175 PTAS
82
mi computer
CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR
Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen VII-Fascículo 82

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, 08008 Barcelona
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-067-2 (tomo 7)
84-85822-82-X (obra completa)
Depósito Legal: B. 52-84

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 078508
Impreso en España-Printed in Spain-Agosto 1985

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 19 425 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 429 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S.A. (Paseo de Gracia, 88, 5.°, 08008 Barcelona), o también con talón bancario remitido a la misma dirección.

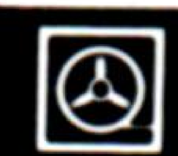
c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

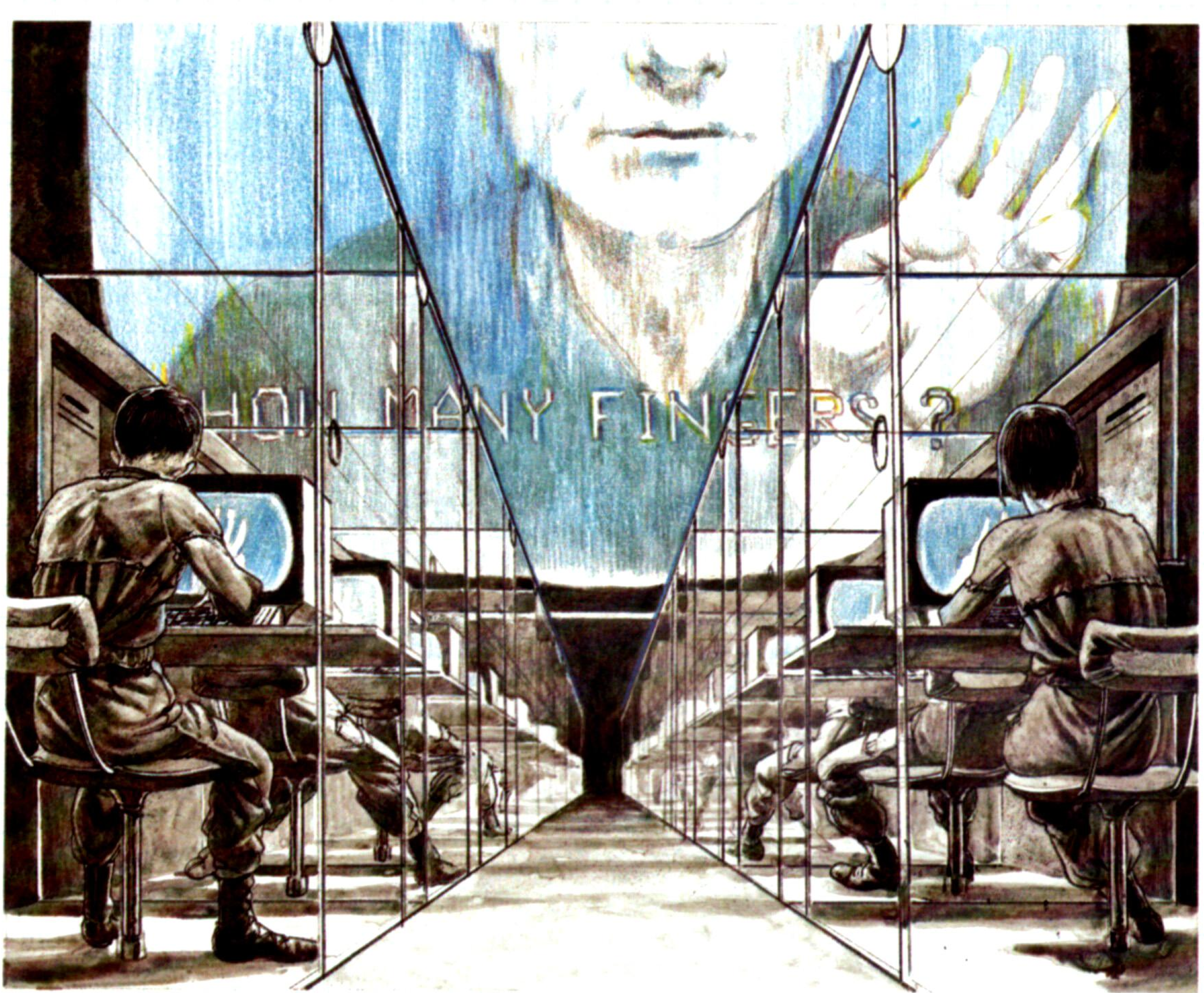
Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.







Nick Harris

Adoctrinamiento interactivo
La imagen del CAL como precursor de una pesadilla orwelliana se debe en gran parte al trabajo de los primeros psicólogos conductistas y muy poco a los desarrollos de hoy en día. La tecnología del disco compacto, el video interactivo y las redes representan grandes promesas para el futuro del aprendizaje asistido por ordenador

¿Retorno a lo básico?

En esta ocasión nos referiremos al CAL y comentaremos las opiniones tanto de sus defensores como de sus detractores

El vocablo CAL corresponde a las siglas de *computer-assisted learning*: aprendizaje asistido por ordenador. También en algunas ocasiones se lo conoce como CAI (instrucción asistida por ordenador) o CBL (aprendizaje basado en ordenador). El término se ha utilizado en contextos notablemente diferentes para definir desde una clase determinada de software didáctico hasta cualquier actividad educativa de carácter general que implique el empleo de ordenadores. En este capítulo hablaremos del CAL en su sentido más estricto, es decir, para describir programas de enseñanza y *no* juegos, simulaciones o programación realizada por el niño. En el CAL el ordenador se utiliza para transmitir información, para constatar si ha sido comprendida, supervisar los ejercicios y las prácticas, y actuar como una máquina "inteligente" de aprendizaje.

En Gran Bretaña, la opinión actual sitúa los programas CAL en la misma línea de actividades educativas tan banales como garabatear en la pizarra o rellenar el tintero. Este bajo concepto acerca del CAL puede decirse que se generó en los primeros días del "aprendizaje programado" y el trabajo de Skinner y sus seguidores. Esta negativa reacción se ha visto reforzada por el lanzamiento, en los años subsiguientes, de programas CAL de escasa calidad, y ahora los maestros observan con gran recelo este aprendizaje. "Los programas CAL que hemos tenido en la escuela han sido pésimos", comenta Alan Coode, quien obtuviera el premio del periódico británico *Sunday Times* a la promoción del alfabetismo informático en la educación. "Se basan en una superada filosofía de la educación y consideran el niño en términos conductistas. Se necesitaría esgrimir algún argumento muy efectivo para que me decidiera a comprar otro programa CAL."

La historia del desarrollo del CAL no es alentadora. A comienzos de 1980, con un parque de menos de 100 microordenadores en las escuelas británicas, era evidente que no había mercado para el software educativo. Los maestros que tenían acceso a los ordenadores debían escribir ellos mismos sus programas en BASIC y, dado que no eran programadores experimentados, su software era, por adjetivarlo de alguna forma, primitivo. Un maestro se pasó días escribiendo un programa que generaba

operaciones de suma, resta, multiplicación y división. Si el niño calculaba la respuesta correcta, en la pantalla aparecía un enorme tilde; si el resultado era incorrecto, se visualizaba una cruz. Los niños que normalmente odiaban "hacer sumas" disfrutaban haciéndolas en el ordenador. El programa obtuvo un enorme éxito e, independientemente de cuán poco imaginativos, amables y plagados de errores fueran los programas, a los niños les resultaba agradable utilizarlos. Lo que los cautivaba era el medio, no el mensaje. Pero a nadie se le ocurrió pensar que los niños estaban empleando un recurso muy caro para una actividad que se podía realizar igualmente con lápiz y papel.

Movidos por el interés de mantenerse a tono con los tiempos, muchos maestros han aceptado programas CAL que, en otras circunstancias, probablemente condenarían. A consecuencia de este entusiasmo mal dirigido y de la manipulación comercial, a comienzos de los años ochenta se produjo un alud de software CAL. En 1982, los programas de enseñanza, incluyendo ejercicios de destreza y práctica, representaban casi el 80 % del *School micro directory*, uno de los principales catálogos de software educativo de Estados Unidos. Esta tendencia experimentó enseguida un retroceso. La oposición que suscitó el CAL la resumió David Chandler en su libro *Young learners and the microcomputer* (Los jóvenes estudiantes y el microordenador): "El microordenador —señala— es una herramienta de asombroso poder que está haciendo posible que la práctica educativa dé un gigantesco paso atrás, al s. XIX. Es la última arma de quienes desean 'retornar a lo básico', permitiéndoles procesar niños de 'entrada' a 'salida' en función de 'objetivos behavioristas' y 'control de calidad...'"

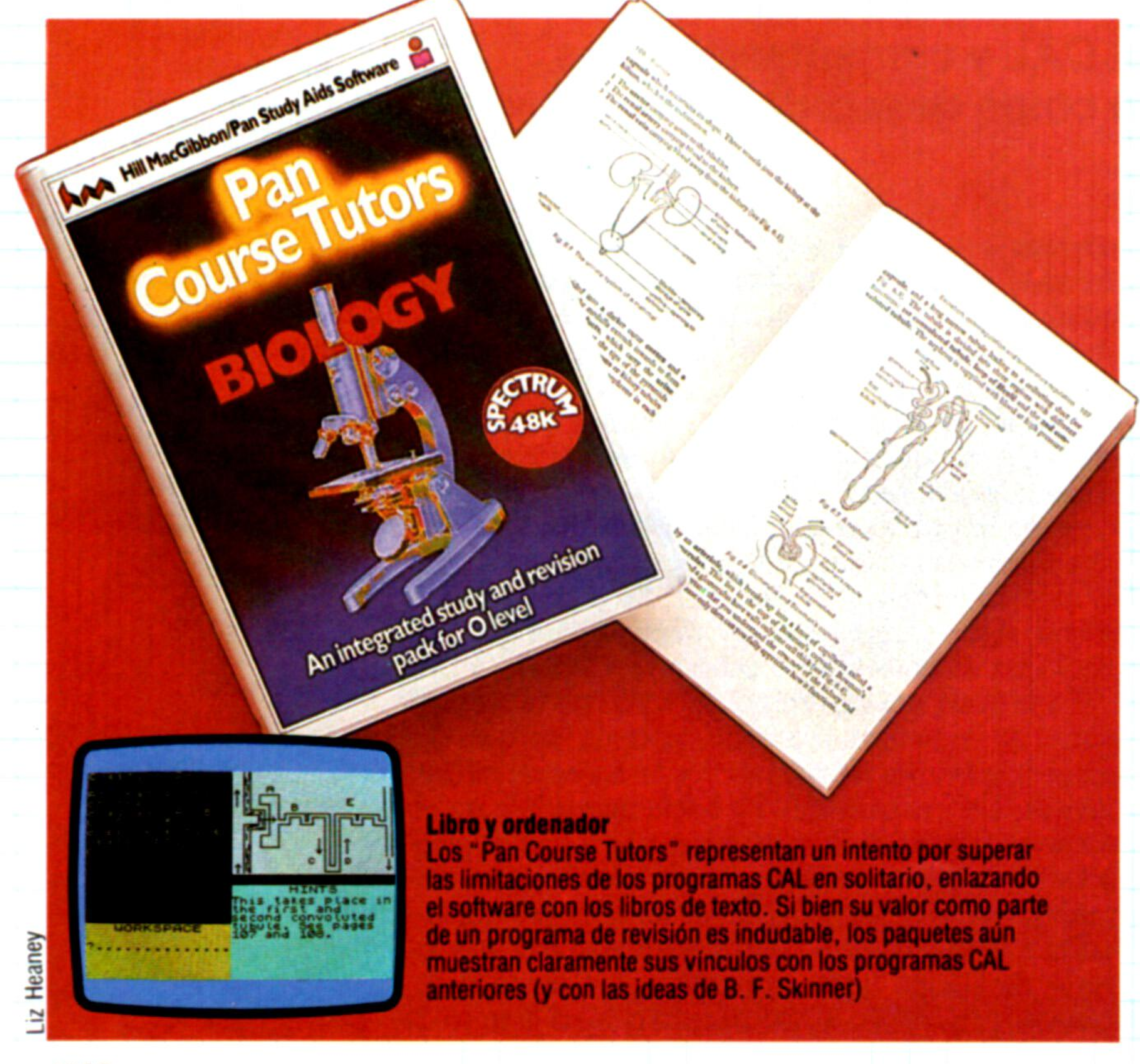
Lamentablemente, gran parte del material CAL ofrece enormes evidencias que apoyan esta condena tan categórica. Muchos programas refuerzan malos hábitos, favorecen respuestas no imaginativas y no dan más opciones que ofrecer una respuesta correcta o incorrecta. Aunque se afirma que el CAL permite que el niño aprenda a su propio ritmo, sería más exacto decir que el niño aprende al ritmo del ordenador.

No obstante, hay signos de un resurgimiento del interés por el aprendizaje asistido por ordenador. La mayor parte de los ordenadores personales los emplean varones de entre 11 y 18 años para practicar juegos de tipo recreativo, y este mercado ha ayudado a establecer una gran base de software que se utiliza ampliamente en el campo de la educación. Como extensión, recientemente ha aparecido un enorme mercado para software CAL, como apoyo al estudio en casa para preparar los exámenes escolares. Una de las principales firmas británicas de software educativo, Hill MacGibbon, ha sabido reconocer este mercado nuevo y ha invertido en la producción de algunos estimables programas CAL de apoyo a los planes de estudio para la preparación de exámenes.

Al igual que otras empresas de software, Hill MacGibbon ha identificado dos áreas principales del software educativo. La primera es la del paquete diseñado para un fin específico, al objeto de preparar al alumno para exámenes académicos o profesionales. La segunda área está relacionada con la educación en su sentido más amplio, en la cual los estudiantes pueden utilizar el ordenador para explorar temas de otro modo inaccesibles.

Un buen ejemplo de la primera categoría de productos es la serie "Pan Course Tutors", editada conjuntamente por Pan Books y Hill MacGibbon. La serie se compone de seis títulos (*Matemáticas, Biología, Química, Economía, Francés* y *Física*) y cada uno viene acompañado por un libro de texto. Ello evita el problema de intentar incluir en el programa grandes cantidades de texto, dejando al software libre para concentrarse en el aspecto interactivo del aprendizaje. El formato de libro/programa muy probablemente persistirá mientras los usuarios de micros personales dispongan de máquinas de ocho bits; pero a medida que se vayan introduciendo en los hogares procesadores capaces de direccionar más memoria, los libros de texto se irán volviendo, en gran medida, innecesarios. La perspectiva más probable es que el software con ventanas permita la presentación de texto y la interacción con el alumno en la misma pantalla.

En la serie "Pan Course Tutors", los programas formulan preguntas sobre la materia elegida. Entre los beneficios del uso del ordenador se incluye la ventaja psicológica de no poder recurrir a la página donde vienen las respuestas y, si se da una respuesta incorrecta, se guía al alumno para hallar la respuesta correcta, con una pista gráfica o un mensaje como "¿Te has acordado de incluir los paréntesis?". Si se adelanta una segunda respuesta equivocada, se visualiza otra pista, por lo general una referencia a una página o sección del libro de texto, para que el estudiante pueda consultar las referencias. Las respuestas se cronometran y se registran, de modo que un programa puede analizar los resultados e incluso sugerir áreas susceptibles de un estudio más profundo. Al trabajar alternativamente con el ordenador y el libro de texto, los estudiantes pueden aprender a su propio ritmo y tomándose el tiempo que necesiten, sin que haya competitividad.



Libro y ordenador
Los "Pan Course Tutors" representan un intento por superar las limitaciones de los programas CAL en solitario, enlazando el software con los libros de texto. Si bien su valor como parte de un programa de revisión es indudable, los paquetes aún muestran claramente sus vínculos con los programas CAL anteriores (y con las ideas de B. F. Skinner)

Liz Heaney



El aprendizaje asistido por ordenador también se puede utilizar para enseñar a los niños a usar el micro como una herramienta. Un programa que guíe al niño a través de las diversas funciones de un procesador de textos, por ejemplo, tiene un enorme valor. Es más fácil seguir las indicaciones en la pantalla, digitando las instrucciones adecuadas y obteniendo realimentación, que ir recorriendo laboriosamente el manual. El Apple Macintosh emplea este tipo de presentación en disco, en unión con una cassette de audio, para enseñar las características del ordenador, el procesador de textos y los programas para gráficos.

Lamentablemente, el crecimiento del mercado potencial de software educativo ya ha suscitado algunas extravagantes consideraciones sobre la pobreza de los productos, lo que, como cabe suponer, inhibe el desarrollo de software de calidad. Una conocida editorial creó un costoso paquete compuesto por cuatro programas, uno de los cuales era un juego de *rounders* (juego de pelota parecido al béisbol). Según la nota que lo acompañaba, éste tenía por objetivo su utilización durante un día de lluvia para enseñar a los niños las reglas del juego. Pero debido a que el programador había tomado tantos atajos al diseñar el paquete (incluyendo la omisión de muchos de los puntos más delicados de las reglas), probablemente los niños que intenten aprender a jugar al *rounders* con este paquete lo tendrán muy difícil. Entre otras afirmaciones excesivas que se hicieron sobre sus beneficios educativos se incluía la siguiente: "El programa ofrece muchísimas oportunidades para la digitación en el teclado... Comienza al pedir a uno de los dos jugadores que entre su nombre." También afirma estimular el interés por los deportes. El programa de juego está escrito en BASIC y es tan lento que incluso carece de la emoción de un videojuego sin valor educativo.

Cuando se agote la novedad que suponen los ordenadores, los maestros y los especialistas en educación considerarán más críticamente el software y expresarán sus necesidades con mayor claridad. Probablemente en el futuro veamos menos programas de aprendizaje asistido por ordenador pero de un estándar más elevado, útiles en áreas específicas de los planes de estudio o para personas que estudien en casa.

La línea divisoria entre lo que se define como aprendizaje asistido por ordenador y el resto de software educativo (con el cual el niño tiene más control sobre lo que está sucediendo) no existe en la realidad. La presentación didáctica se puede combinar con simulaciones y experimentos que, de otra forma, sería imposible llevar a cabo en el laboratorio de la escuela.

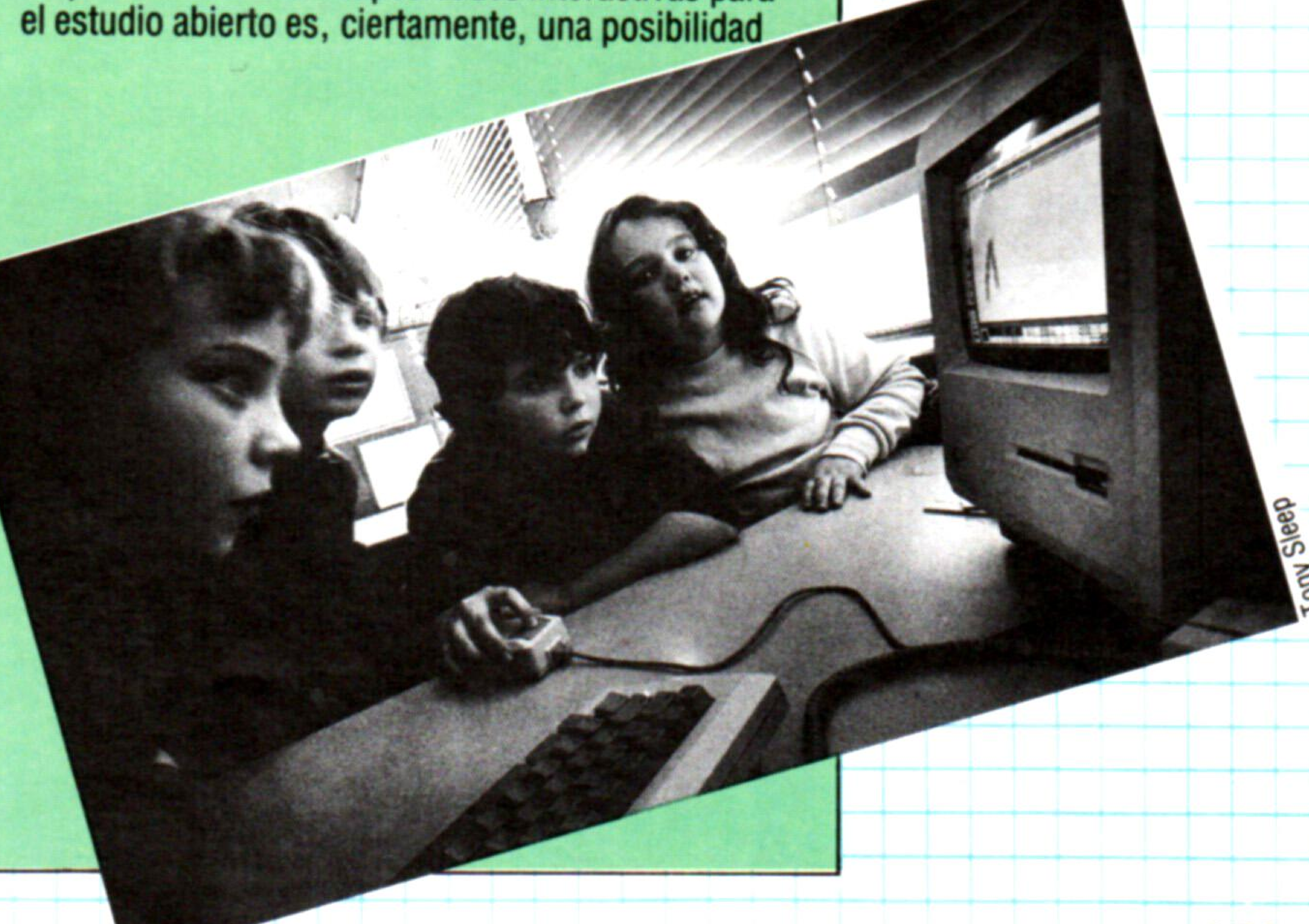
Simular la realidad

El software CAL actual es útil para enseñar lo que miden los "tests" estandarizados, pero este software aporta muy poco para que el educando ponga en juego su imaginación. Para que el CAL del futuro sea un medio auxiliar atractivo y valioso para la educación, es imprescindible que cambien tres factores. Es preciso descartar las técnicas educativas anticuadas, como el aprendizaje en base a respuestas correctas, en el que se fundamentan tantos programas CAL. Se necesitan ordenadores más potentes; los microordenadores actuales, con su limitada memoria, no pueden soportar la gama de software o potencia necesaria para el aprendizaje interactivo asistido por ordenador. Pero el factor más restrictivo ha sido que el diseño de los programas ha estado en manos de quienes comprenden la programación, que no son necesariamente quienes mejor saben cómo aprende la gente.

Un primer intento por cambiar esta situación fue el PILOT, un lenguaje para ordenador diseñado para permitir a los maestros la creación de paquetes de aprendizaje para niños. En el Centro de Investigación de Xerox de Palo Alto, California, se han continuado los trabajos en esta área. Allí se ha desarrollado un Rehearsal World (mundo de ensayo) utilizando un entorno de programación visual. Fueron los trabajos llevados a cabo en este laboratorio los que llevaron al desarrollo del Apple Macintosh, y Rehearsal World parece ampliar el concepto que subyace tras la máquina. Quienes no sean programadores pueden utilizar Rehearsal World para producir software educativo en SMALLTALK. El diseñador mueve "actores" a través de un "escenario", enseñándoles cómo interactuar enviándoles "pies", todo ello en un entorno gráfico interactivo. Se intenta utilizar este entorno de un modo similar a los gráficos de tortuga del LOGO. Rehearsal World necesita muchísima memoria, pero marca una de las direcciones que seguirá el CAL en el futuro.

Incluso en la actualidad, un ordenador ofrece al alumno muchas rutas a través de un programa de aprendizaje, y las posibilidades potenciales del empleo del video interactivo son enormes. La combinación de un lenguaje creativo como el MICROTEX o el Apple SUPER PILOT con el video interactivo proporcionaría al educador un control sin precedentes sobre la situación de aprendizaje.

En la escuela, el video interactivo se puede utilizar para impartir clases, proporcionando una fuente de sonido e imagen, o bien lo pueden usar grupos reducidos de niños que empleen el disco como preceptor. El aprendizaje individualizado empleando todas las capacidades interactivas para el estudio abierto es, ciertamente, una posibilidad


Tony Sleep

En plena actuación
La difusión de nuevos sistemas operativos de "entorno gráfico", como el Macintosh de Apple (abajo) y el GEM de Digital Research está poniendo un considerable poder de proceso bajo el control de usuarios que anteriormente eran analfabetos desde el punto de vista informático. Las implicaciones de las nuevas máquinas "amables con el usuario" respecto al aprendizaje asistido por ordenador son enormes, porque anteriormente todos los programas CAL debían trabajar sobre el supuesto de que el usuario no tenía ningún control sobre la máquina como no fuera responder a una pregunta cuando así se le pidiera

Hombre al agua

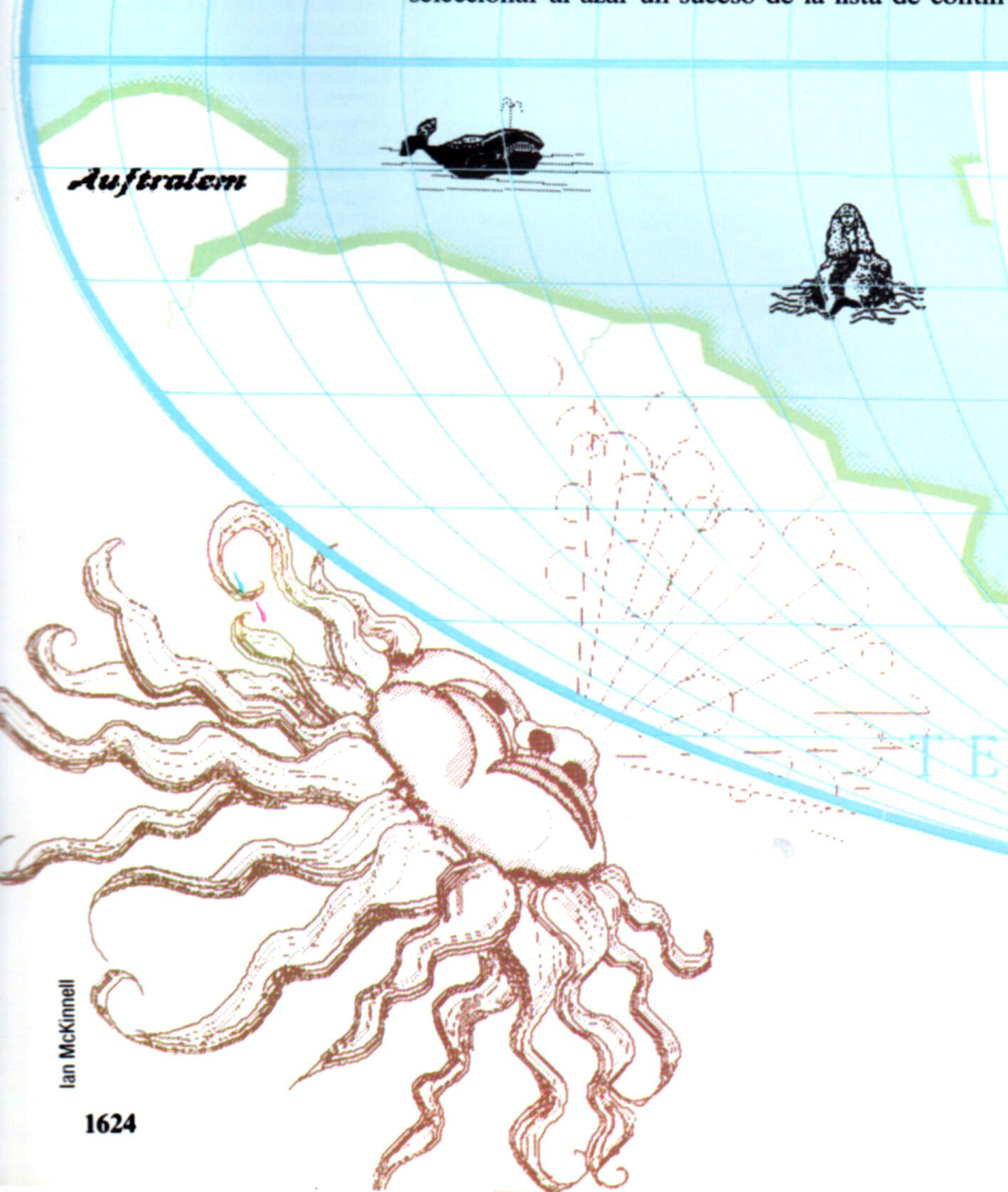
El Módulo 6 de nuestro juego de simulación se ocupa de algunos de los sucesos y vicisitudes a los que puede verse enfrentada la tripulación durante la larga travesía del océano

Para poder simular los sucesos y vicisitudes que ocurrían durante los largos viajes marítimos del s. XVI, debemos preparar el programa para que produzca un cierto número de eventos. Algunos de estos eventos pueden ser de ayuda para los marineros, como que sople un viento favorable, pero otros pueden tener efectos nefastos para la travesía, como sería que algunos miembros de la tripulación o algunas provisiones se cayeran al agua.

Hay un total de 16 eventos posibles, sólo dos de los cuales se pueden producir en una semana dada. En este capítulo escribiremos una subrutina para seleccionar al azar un suceso de la lista de contingencias, incluyendo el código para las cinco primeras contingencias. Anteriormente habíamos insertado un trozo de código para establecer al azar las tasas de fortaleza de la tripulación. Antes de digitar el Módulo 6, suprima las líneas de la 601 a la 604. El nuevo código se compone de varias partes: una sección de inicialización, dos líneas en el bucle principal del viaje y una subrutina para seleccionar un evento al azar y manejar cinco de éstos.

Es importante que cada evento aleatorio se produzca una sola vez durante el viaje. Para conseguirlo, en la línea 42 se DIMensiona una matriz, RR(), que contiene 16 elementos que corresponden a los 16 eventos posibles. Cada elemento se establece en 0. Cuando se produzca un evento, el elemento correspondiente se establecerá en 1, de modo que el programa lo reconocerá y no lo repetirá. Las variables que indican coordenadas dentro de un programa se denominan *flags* o *banderas*. Es necesario llevar la cuenta de los eventos a medida que se vayan produciendo para evitar que el programa busque incesantemente otro evento en la matriz, una vez que todos ellos se hayan producido. La cantidad de eventos se registra en la variable RC, que se incrementa cada vez que se genera uno. El número de posibles eventos está retenido en RM. Observe que, como en el próximo capítulo iremos añadiendo más eventos, en el futuro será necesario cambiar el valor de RM para hacerlo corresponder a la nueva cantidad.

En el bucle principal del viaje, la línea 860 llama a la subrutina de la línea 5500, que genera un evento al azar. Esta llamada está insertada en el bucle principal de modo que el evento se genere después de que el Diario de Navegación del Capitán haya dado la duración estimada del viaje. La subrutina 5500 selecciona cuál de los eventos se producirá cada semana. Si ya han acontecido todos, el juego retorna al programa principal sin generar ningún evento. La línea 5502 comprueba esto mediante la comparación de RC, el número de eventos que ya se han producido, con RM, el número de eventos del programa.



Ian McKinnell

Eventos aleatorios

En la línea 5510 se genera un número al azar entre 1 y el número de eventos posibles, RM. La línea 5520 comprueba en la matriz RR() si el elemento para ese evento se ha establecido en 1, lo que significa que ya se ha producido. De ser así, retorna al programa principal sin generar ningún evento. Cuando se ha seleccionado un evento nuevo, la línea 5523 establece en 1 la bandera correspondiente en la matriz RR(), para impedir que el mismo vuelva a ser seleccionado. La misma línea incrementa en uno el contador de acontecimientos, RC.

En la línea 5525 se utiliza la sentencia ON X GOTO para dirigir el programa al código apropiado para el evento seleccionado. ON...GOTO y ON...GOSUB son,

juntas, una forma de reemplazar un número de sentencias IF...THEN. Cada instrucción va seguida por una lista de números de línea. Si la variable es 1, se selecciona el primer número de línea de la lista. Si su valor es 2, entonces se selecciona el segundo número de línea, y así sucesivamente. Observe que el Spectrum carece de ON...GOTO y ON...GOSUB. Para este módulo habrá de remitirse al recuadro *Complementos al* BASIC.

Tenemos cinco sentencias correspondientes a los cinco eventos que estamos cubriendo. El primer evento, una persona que es arrastrada por la borda, se produce en la línea 5540. Los últimos cuatro números de la lista ON...GOTO son los mismos, de modo que tanto si el número aleatorio es 2, 3, 4 o 5, se envía el programa al mismo sitio. Las cuatro contingencias de los elementos del 2 al 5 consisten en provisiones que caen por la borda, y las cubre el mismo trozo de programa que comienza en la línea 5570.

Si se selecciona el primer número de línea, se imprime un mensaje que informa que una persona ha caído al agua. La línea 5554 imprime el nuevo total de la tripulación, restando 1 a CN, la variable que representa el número de tripulantes, pero no altera el valor de CN en esta etapa. Ello se hace durante el informe de final de semana, comprobando si la tasa de fortaleza de algún miembro se ha establecido en −999.

Por supuesto, debe elegirse qué tripulante caerá al agua. La selección se efectúa en el bucle de la línea 5558, que busca en la matriz de categoría/fortaleza de la tripulación, desde el principio, ignorando todos los elementos establecidos en 0 o −999, y seleccionando como víctima al primer tripulante vivo. La línea 5560 arroja éste al mar, estableciendo su tasa de fortaleza en −999. El valor de T se establece entonces en 16, para hacer que el programa salga del bucle.

Los restantes eventos de los que nos ocupamos en este módulo, etiquetados del 2 al 5 en la matriz RR(), tratan de los cuatro tipos de provisiones que caen al agua. Para hacer más interesante el juego, se selecciona una cantidad aleatoria de cada provisión para lanzar por la borda. La cantidad de fruta, verduras, carne y agua se representa mediante PA(1), PA(2), PA(3), PA(4).

En la línea 5572 se le resta 1 al número aleatorio, X, generado por la subrutina 5500, ya que al perder un tripulante ya se ha cubierto un evento. El número aleatorio estará ahora entre 1 y 4, correspondiendo a uno de los cuatro tipos de provisiones. La línea 5574 comprueba si la provisión está agotada y, si así es, no emprende ninguna acción. El programa informa, entonces, al jugador que alguna de las provisiones se ha ido por la borda.

La cantidad de comida o agua perdida puede ser una tercera, cuarta o quinta parte de las existencias. La fracción se elige al azar en la línea 5592, dividiendo la cantidad de provisiones restantes, PA(), por tres, cuatro o cinco, y restando esa fracción a la cantidad original. El programa calcula luego la cantidad de raciones semanales que quedan de esa provisión restante, mediante la ecuación de la línea 5594.

Se puede ejecutar el programa con las cinco contingencias ya mencionadas. Todas ellas tienen un efecto negativo sobre el viaje, poniendo más dificultades para llevarlo a cabo con éxito.

Módulo 6: Eventos al azar

Inicializar variables eventos

```
42 DIM RR(16)
43 REM INDICADORES PARA MOSTRAR SI YA HA OCURRIDO EL
   EVENTO AZAR(N)
44 RC=0
45 REM CONTADOR EVENTO AL AZAR HASTA AHORA
46 RM=5
47 REM NUM. DE EVENTOS ALEATORIOS DEL PROGRAMA
```

Adición al bucle principal del viaje

```
860 GOSUB 5500
861 REM GO TO GENERAR EVENTO AL AZAR
```

Subrutina de eventos aleatorios

```
5500 REM GENERADOR DE EVENTOS AL AZAR
5502 IF RC=RM THEN RETURN
5503 REM SALIR SI TODOS LOS EVENTOS YA PRODUCIDOS
5504 PRINTCHR$(147)
5505 GOSUB 9200
5510 X=INT(RND(1)*RM)+1
5515 REM GENERAR NUM. AL AZAR ENTRE 1 y RM
5520 IF RR(X)=1 THEN RETURN
5222 REM RETURN SI ESTE EVENTO YA SE PRODUJO
5523 RR(X)=1:RC=RC+1
5524 REM ESTABLECER INDIC. PARA SEÑALAR QUE ESTE EVENTO
     YA SE PRODUJO E INCREMENTAR CONTADOR EVENTOS
5525 ON X GOTO 5540,5570,5570,5570,5570
5528 REM IR A CODIGO APROPIADO PARA ESTE EVENTO
5530 PRINT:S$=K$:GOSUB 9100
5535 GET I$:IF I$=""THEN 5535
5539 RETURN:REM RETORNAR A BUCLE PRINCIPAL VIAJE
5540 REM EVENTO 1 – HOMBRE AL AGUA!
5542 PRINT
5543 S$="  DURANTE LA SEMANA*":GOSUB 9100
5545 PRINT:GOSUB 9200
5546 S$="   1 PERSONA CAYO POR LA BORDA*":GOSUB 9100
5548 S$="   DURANTE UNA TORMENTA*":GOSUB 9100
5550 PRINT:GOSUB 9200
5552 S$="TU TRIPULACION SE HA REDUCIDO AHORA A*":
     GOSUB 9100
5554 PRINT CN-1;"MIEMBROS"
5558 FOR T=1 TO 16
5559 REM BUSCAR QUE TRIPULANTE SE PERDIO
5560 IF TS(T,2)=0 OR TS(T,2)=-999 THEN 5566
5562 TS(T,2)=-999:REM MUERTO
5564 T=16
5566 NEXT
5568 GOTO 5530
5570 REM EVENTOS DEL 2 AL 5 – PERDIDA DE PROVISIONES
5572 X=X-1:REM AHORA X INDICA LA PROVISION (1-4)
5574 IF PA(X)=0 OR PA(X)=-999 THEN 5530
5576 REM NINGUNA ACCION SI YA SE HA AGOTADO ESTA
     PROVISION
5578 PRINT
5580 S$="   DURANTE LA SEMANA*":GOSUB 9100
5582 PRINT:GOSUB 9200
5584 PRINT"   PARTE DE TU  ";P$(X)
5586 S$="   FUE ARRASTRADA POR LA BORDA*":GOSUB 9100
5588 PRINT:GOSUB 9200
5590 S$="AHORA TIENES APROXIMADAMENTE*":GOSUB 9100
5592 PA(X)=PA(X)-INT(PA(X)/(INT(RND(1)*3)+3))
5593 REM REDUCIR CANTIDAD PROV EN 1/2 1/3 o 1/4
5594 PRINT INT(PA(X)/(CN*PN(X)));
5595 PRINT "Y TE QUEDA PARA  ";P$(X);"   SEMANAS
     APROXIMADAMENTE"
5599 GOTO 5530
```

Complementos al BASIC

Spectrum:
Efectuar las siguientes modificaciones:

```
5504 CLS
5525 IF X=1 THEN GOTO 5540
5526 IF X>1 AND X<6 THEN GOTO 5570
5535 LET I$=INKEY$:IF I$=""THEN GOTO
     5535
```

BBC Micro:
Efectuar las siguientes modificaciones:

```
5504 CLS
5535 I$=GET$
```

Capturando "klingons" en 17 dimensiones

Hablaremos del uso de las matrices, incluyendo aspectos tales como empaquetamiento, índices, etc.

En muchos lenguajes de programación la matriz constituye un medio muy natural de asociar datos del mundo real tales como listas, tablas y matrices a una estructura de datos para ordenador, y una forma eficaz de facilitar el acceso a elementos individuales para su proceso. No obstante, la principal causa de su uso casi universal es producto sólo del hábito. Sencillamente, la mayor parte de los lenguajes primitivos no disponían de ningún método de estructuración de datos aparte de la matriz. El primero de los lenguajes de alto nivel, el FORTRAN, sólo tenía números complejos y matrices, y su descendiente, el BASIC, omitió los números complejos. Éste es el motivo por el cual el bucle FOR...NEXT es la única estructura de iteración definida.

Todo ello estaba muy bien en los días en que el FORTRAN se empleaba sólo para efectuar operaciones de matrices sobre todos los elementos de una matriz, y cuando el BASIC se utilizaba sólo para introducir a los estudiantes en los rigores de la programación en FORTRAN. Ya hemos visto algunos ejemplos de la flexibilidad extra de control que le confieren al PASCAL los bucles activados por condición (WHILE y REPEAT). Y, lo que quizá sea aún más importante, estamos comenzando a vislumbrar parte del inmenso potencial y sencillez de expresión que ofrecen las otras estructuras de datos del PASCAL, como los conjuntos y los registros. En PASCAL, por consiguiente, el dominio de la matriz como una "herramienta de datos" universal se ve considerablemente reducido.

Matrices flexibles

Si bien la mayoría de los programadores suelen sentirse en terreno conocido cuando se enfrentan con las matrices del PASCAL, hay dos puntos significativos que se han de tener siempre presentes. Debido a que el PASCAL posibilita una elección tan amplia de los métodos para estructuración de datos, muchos problemas de programación se pueden muy bien resolver con más eficacia mediante el empleo de otras estructuras. En segundo lugar, las matrices no están limitadas por ninguna restricción en cuanto a la complejidad estructural de sus elementos, de modo que en PASCAL se puede obtener una flexibilidad mucho mayor.

La definición de un tipo de matriz reserva espacio de almacenamiento para un cierto tamaño fijo de matriz, y define tanto el tipo base de su índice como el tipo de cada componente de la misma. De modo que, por ejemplo:

```
TYPE
   ContCaract=ARRAY['A'..'Z'] OF integer;
VAR
   lista : ContCaract;
```

reservaría almacenamiento para 26 enteros que serían referenciados especificando el identificador de matriz seguido por una expresión de indexación entre paréntesis. Con las matrices siempre se utilizan corchetes, tal como se hacía originalmente en BASIC (el posterior empleo de los paréntesis para el subíndice se debió sólo al hecho de que algunos juegos de caracteres limitados carecían de corchetes). Para acceder a un determinado entero de la lista que acabamos de mencionar, podemos escribir:

lista ['M'] o lista [pred(simbolo)]

En el segundo ejemplo, la expresión pred (simbolo), por supuesto, debe tener un valor char comprendido entre el subrango de la 'A' a la 'Z'. Para indexar una dimensión de matriz se puede emplear cualquier tipo escalar, pero no tipos estructurados ni reales. Esto impide tentativas absurdas de referirse a lista ['segundo'] o bandera[3.74], por ejemplo. Utilizando la definición de tipo ilustrada, podemos inicializar a cero todos los elementos del contador de una matriz de tipo ContCaract mediante:

```
VAR
     letra        :char;
     contador     :ContCaract;
BEGIN
   FOR letra :='A' TO 'Z' DO
     contador [letra]:=0
```

y así sucesivamente. Obviamente, contador[1] es ilegal para esta estructura (es un tipo de índice incorrecto) y contador ['a'] no existe (el subíndice está fuera del subrango definido), de manera que es una buena costumbre definir identificadores de tipo para las variables de indexación. En cualquier caso, nos encontraremos invariablemente con que los necesitamos cuando definimos nuestros propios procedimientos y funciones.

Si, a su vez, estos subrangos están definidos con valores dados en una definición CONST, sería posible reescribir todo el programa para tratar con estructuras de distinto tamaño, mediante la alteración de una sola línea. Por ejemplo, si bien en PASCAL no existe ningún tipo serie predeclarado, una forma de producirlo (aunque no necesariamente la mejor) sería:

```
CONST
   LongSerie=25;
TYPE
   TamañoSerie=1..LongSerie;
   serie=PACKED ARRAY [TamañoSerie] OF char;
```



Tipos de datos del PASCAL

Simple {sólo UN valor}
REAL
Escalar
INTEGER
CHAR
BOOLEAN
Subrango
Enumerado
Puntero {dirección}
Estructurado {más de un valor}
Elemental {longitud finita}
Base simple
Base mixta
Matrices
Conjuntos
Registros
Variantes
Avanzado {longitud potencialmente infinita}
TEXT
Otros archivos
Estructuras dinámicas {usando tipos de puntero}

Liz Dixon

¡Típico!
La disciplina estructuración de datos del PASCAL obliga al programador a adoptar un enfoque más metódico en la construcción del programa. Sólo dos de los tres tipos de datos básicos, como vemos en el diagrama, se pueden subdividir en otras categorías. Pero la gama de posibles datos estructurados y simples exige definiciones estrictas de todos los datos que se utilizan en un programa. Con frecuencia, este esfuerzo adicional ofrece al programador de PASCAL la recompensa de soluciones más elegantes a los problemas de la programación que las que pueden proporcionar otros lenguajes

Observe que la palabra reservada PACKED puede preceder a una palabra reservada de cualquier tipo estructurado (SET, ARRAY, RECORD o FILE). Al "empaquetar" estructuras de datos, podemos obtener importantes beneficios en el empleo de memoria, especialmente en ordenadores con grandes tamaños de palabra (16/32 bits o más).

El tipo "serie"

El único momento en el que uno necesita verdaderamente ser consciente del empaquetamiento es cuando se emplean constantes literales en serie. Una serie de caracteres encerrada entre comillas se supone indexada a partir del uno (no del cero) y, por consiguiente, queda declarada implícitamente como:

```
PACKED ARRAY[1..N] OF char
```

donde N es el número de caracteres de la serie. Analicemos este programa:

```
PROGRAM SerieEmpaq;
CONST
  Pascal='Pascal';
TYPE
  serie=PACKED ARRAY[1..10] OF char;
VAR
  S         :serie;
BEGIN
  S:=Pascal;
  S:='Demasiados caracteres';
  S:='Pascal    '
END.
```

En el cuerpo del programa, las dos primeras sentencias son ilegales debido a la incompatibilidad de longitudes, pero la tercera es aceptable (utilizándose cuatro espacios como caracteres de relleno). Si la definición del tipo serie no estuviera empaquetada, todas las asignaciones de literales serían incompatibles. En la práctica, estas restricciones no constituyen un problema serio. Aunque oficialmente el PASCAL sólo soporta el uso de los procedimientos read y write para E/S de estructuras completas desde y hacia archivos del almacenamiento externo, podemos escribir tipos en serie de variables o literales como éstos directamente en la VDU. Intente escri-

bir un programa utilizando una declaración de serie similar a la del último ejemplo, e incorpore la sentencia:

```
REPEAT
   write ('Serie(Q para salir):');
   ReadLn (S);
   WriteLn ('La serie era:"',s,'"')
UNTIL S[1] IN ('Q','q')
```

Una vez que logre ponerlo en funcionamiento, pruebe el programa e investigue los siguientes puntos:

- ¿Se ignoran los espacios delanteros o tabulaciones?
- ¿Qué ocurre cuando la línea es más larga que la longitud de la serie?
- ¿Qué contiene S cuando se entra un solo carácter?
- ¿Qué ocurre si sólo se pulsa Return?
- ¿Puede usted incluir un código adecuado para rellenar con espacios los elementos sin importancia?

Por cuanto concierne al PASCAL, no existe ningún límite en absoluto respecto al tamaño ni al número de dimensiones que uno puede especificar para una matriz. Si usted puede imaginar cómo se capturarían los *klingons* en un medio continuo de espacio-tiempo en 17 dimensiones, ¡entonces puede crear una estructura de datos adecuada en PASCAL! Sin embargo, habrá de disculpar a su ordenador si no recibe con agrado estas definiciones de tipos:

```
GranTipo=ARRAY [integer] OF SET OF char;
   {una demanda de al menos 64 K×128 bytes!}
AunMayor=ARRAY[1..1000] OF
      RECORD
         apellido,
         nombre : serie;
         direccion : ARRAY [1..5] OF serie;
         presente : SET OF asistencias;
         {etc.}
   END:{AunMayor}
```

El tamaño físico de la memoria limitará de modo inevitable una programación tan ambiciosa. Nuevamente ello refuerza el valor de poder especificar subrangos de los tipos escalares, minimizando así toda sobrecarga de almacenamiento.

La criba de Eratóstenes

Este programa muy conocido para hallar números primos se optimiza, desde el punto de vista de velocidad de ejecución, ignorando los números pares y utilizando una matriz de "banderas" (8 192 elementos para primos hasta 16 384). Por tanto, se requieren al menos 8 K de memoria, quizá 32 K en lenguajes que carezcan de variables booleanas de un solo byte y enteras de cuatro bytes. Por este motivo no se ejecutará sin modificar en BASIC BBC. En PASCAL, sin embargo, el conjunto completo requeriría sólo 16 384 bits (2 K) y nos permite reproducir el algoritmo original de Eratóstenes con mayor claridad y precisión:

```
Colocar todos los números (1...max) en una
   criba
Sacar el número uno (primero por definición)
{imprimir si fuera necesario}
REPEAT
   Sacar el número más bajo de la criba
   {imprimir este primo}
   extraer todos los múltiplos de la criba
UNTIL la criba quede vacía
```

Sólo en potentes ordenadores centrales se suele disponer de un conjunto tan grande, pero podemos simular uno mediante el empleo de una matriz de conjuntos más pequeños (100 o 1 000 elementos, supongamos, según su compilador). El índice de la matriz de cada conjunto se halla a partir de la división entera del número, y su pertenencia al conjunto se representa mediante N módulo 100 o 1 000. Sin embargo, a menos que posea un ordenador de 100 bits, probablemente la velocidad será inferior que en la versión de matriz

```
PROGRAM      ListaCriba   (output);
{Hallar numeros primos por algoritmo de Eratostenes}
CONST
  Tamaño Conj          =    100; {~implementacion}
  PredTamañoConj       =     99; {TamañoConj-1}
  Primo Max            =  16383; {para MaxInts=32767}
  MaxLista             =    163; {PrimoMax DIV TamañoConj}

TYPE
  GamaPrimo            = 1..PrimoMax;
  dimension            = 0..MaxLista;
  GamaConj             = 0..PredTamañoConj;
  Criba                = SET OF GamaConj;
  Eratostenes          = ARRAY [dimension] OF Criba;
  cardinal             = 0..MaxInt;

VAR
  Cribas               : Eratostenes;
  contador,
  multiplo             : cardinal;
  indice               : dimension;
  N                    : GamaPrimo;
  numero               : 0..PredTamañoConj;

BEGIN
  WriteLn:
  WriteLn    ('Criba de Eratostenes': 50);
  WriteLn    ('============' : 50);
  WriteLn;
  WriteLn;

  FOR indice := 0 TO MaxLista DO
    Cribas [indice] := [0..PredTamañoConj];
        {poner todos los numeros en las cribas}

  Cribas [0] := [2..PredTamañoConj);
  {WriteLn (1);} {numero primo por definicion}
  contador := 1;

  FOR N := 2 TO PrimoMax DO
    BEGIN
      indice := N DIV TamañoConj;
      numero := N MOD TamañoConj;

      IF numero IN Cribas [indice] THEN
        BEGIN
          contador := succ (contador);
        {WriteLn (N);}
          Cribas [indice] := Cribas [indice]
          - [numero];
          multiplo := N+N;

          WHILE multiplo <= PrimoMax DO
            BEGIN
              indice:=multiplo DIV TamañoConj;
              Cribas [indice] := Cribas [indice]
              -[multiplo MOD TamañoConj];
              multiplo := multiplo+N
            END
        END
    END;

  WriteLn;
  WriteLn (contado( : 25,'primos hallados.')

END.
```

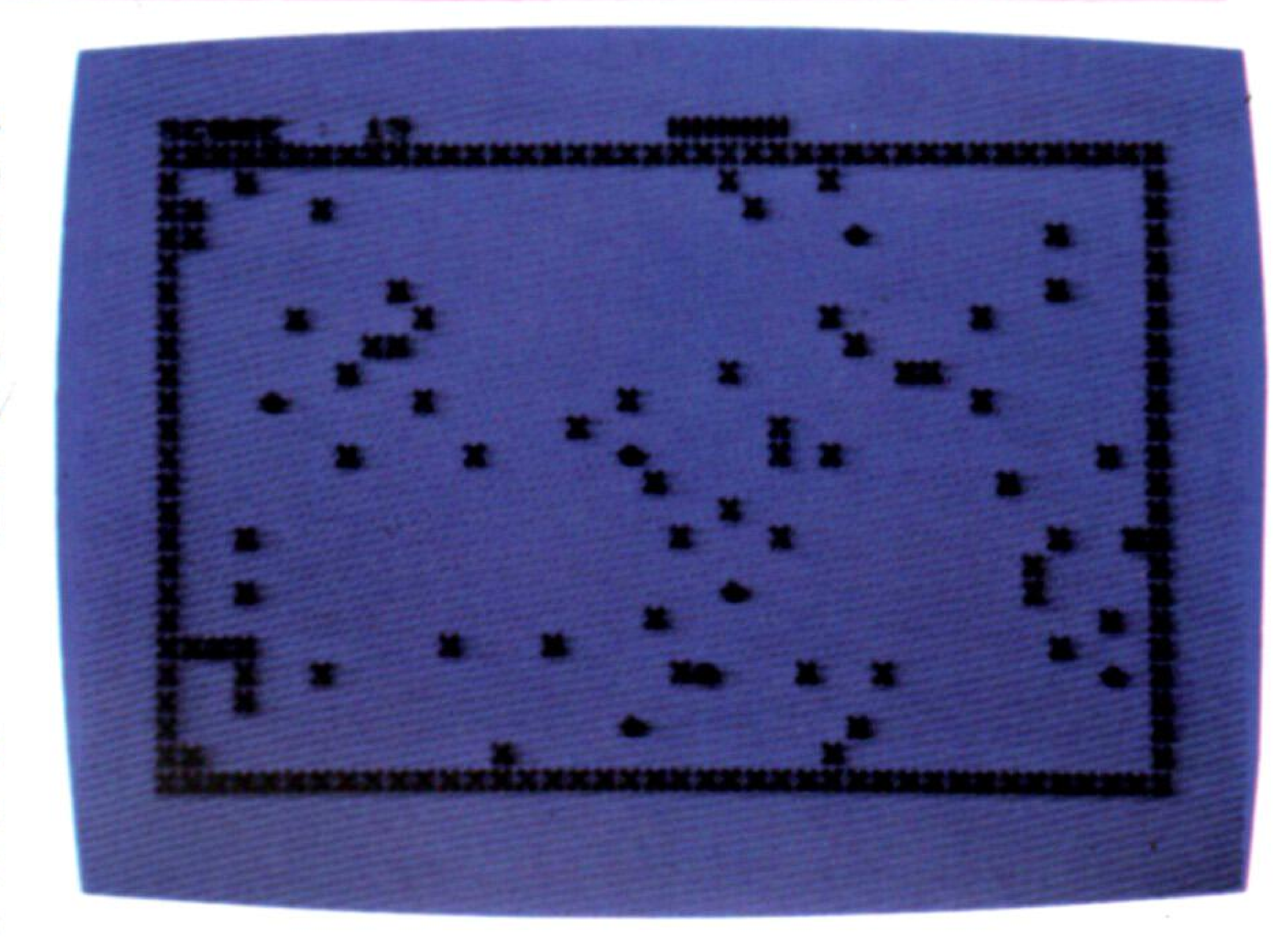
Robots en el C64

Usted está solo, abandonado en un planeta defendido por robots asesinos... Se trata de "Robots", en versión para el Commodore 64

Las minas se hallan representadas sobre la pantalla mediante unas "X". Al comienzo del juego, cinco robots (representados por rombos) se hallan sobre el terreno. Sin perder un segundo, se precipitan hacia usted, siguiendo siempre el camino más corto. Por suerte, los robots son ciegos y no ven las minas que se hallan entre ellos y usted, lo cual le permite eliminarlos, siempre que se desplace del modo adecuado. Para ello debe usar las teclas:

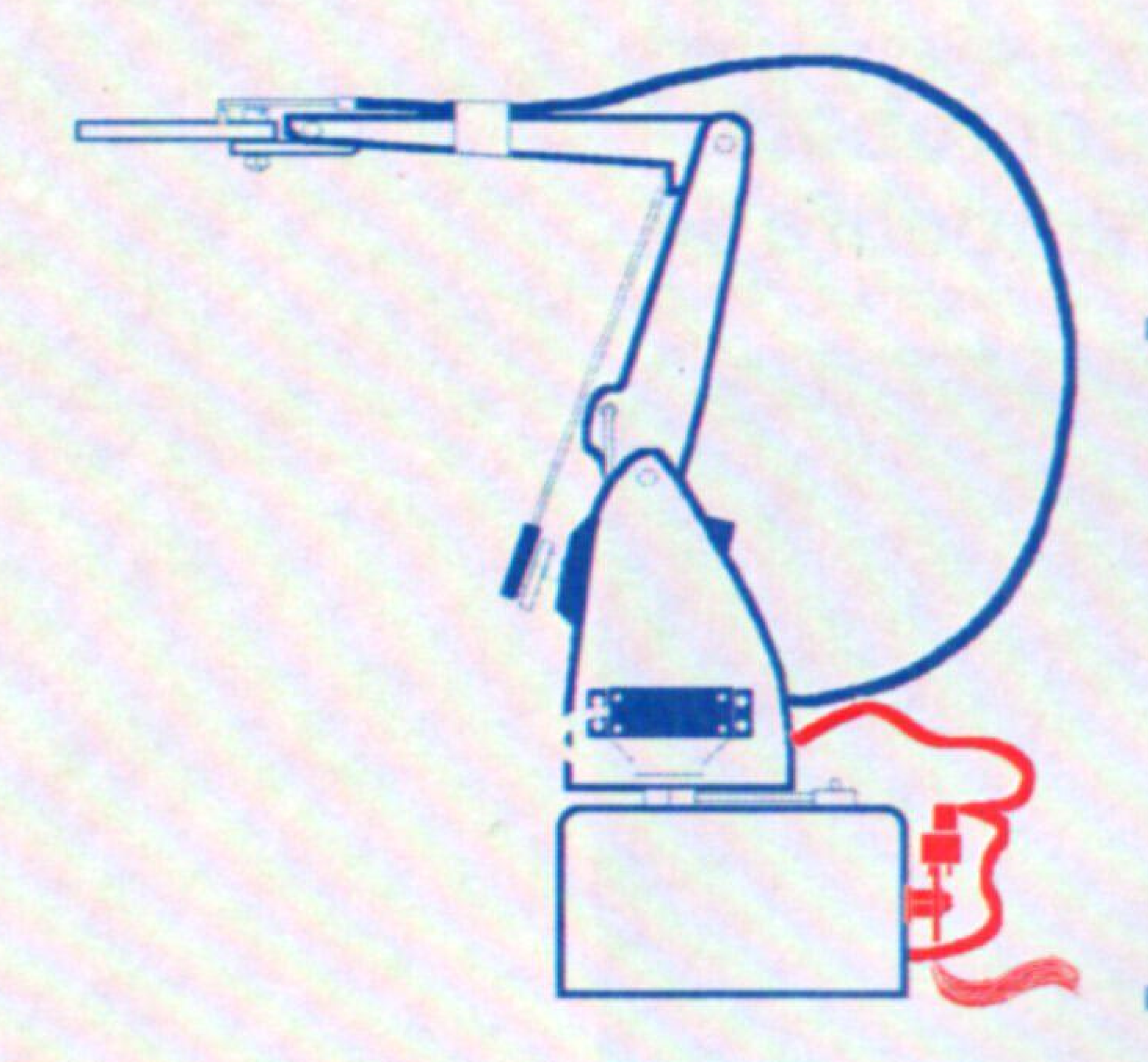
Q, W, E, A, S, D, Z, X, C

según la dirección que haya escogido. La tecla S sirve para detenerse. Cuando haya eliminado a todos los robots, el juego continúa con un robot suplementario. Si salta sobre una de las minas o le mata uno de los robots, aún no se ha perdido todo. En realidad tiene cinco vidas. Si desea cambiar el número de minas, modifique el valor de la variable NM en la línea 1270.



```
5 REM *************
10 REM *   ROBOTS   *
15 REM *************
20 DIM R(30)
25 NH=5
30 GOSUB 1000
100 GET X$
110 GOSUB 900
120 J=J+D
130 C=PEEK(J)
140 IF C<>CN AND C<>CJ THEN 600
150 POKE J1,CN
160 POKE J,CJ
170 POKE J+M,JC
180 J1=J
200 JY=INT((J-1024)/40)
210 JX=(J-1024)-JY*40
220 T=0
230 FOR I=1 TO NR
240 IF R(I)=0 THEN 380
250 T=1
260 RY=INT((R(I)-1024)/40)
270 RX=(R(I)-1024)-RY*40
280 R1=RY+SGN(JY-RY)
290 R2=RX+SGN(JX-RX)
300 RR=40*R1+R2+1024
310 C=PEEK(RR)
320 IF C=CR OR C=CM THEN S=S+1:POKE R(I),
    CN:R(I)=0:GOTO 380
330 IF C=CJ THEN 600
340 POKE R(I),CN
350 POKE RR,CR
360 POKE RR+M,RC
370 R(I)=RR
380 NEXT I
390 IF T=0 THEN 500
400 FOR I=1 TO 500
410 NEXT I
420 GOTO 100
500 S=S+10
510 IF NR<30 THEN NR=NR+1
520 GOSUB 1030
530 GOTO 100
600 NH=NH-1
610 POKE J,CN
620 FOR I=1 TO 4
630 POKE 53281,0
640 FOR K=1 TO 50
650 NEXT K
660 POKE 53281,5
670 FOR K=1 TO 50
680 NEXT K
690 NEXT I
700 IF NH>0 THEN NZ=N1:GOTO 30
710 IF S>RE THEN RE=S
720 PRINT CHR$(147)
730 FOR I=1 TO 4
740 PRINT
750 NEXT I
760 PRINT TAB(13)"PUNTOS[1SPC]:";
775 PRINT S
780 FOR I=1 TO 4
785 PRINT
790 NEXT I
795 PRINT TAB(11)"PUNTUACION[1SPC]MAXIMA [1SPC]:";
800 PRINT RE;
805 FOR I=1 TO 4
810 GET X$
815 PRINT
820 NEXT I
825 PRINT TAB(13)"OTRA[1SPC]?"
830 GET X$
835 IF X$="" THEN 830
840 IF X$<>"N" THEN S=0 : GOTO 25
845 END
900 IF X$="Q" THEN D=-41
910 IF X$="W" THEN D=-40
920 IF X$="E" THEN D=-39
930 IF X$="A" THEN D=-1
940 IF X$="S" THEN D=0
950 IF X$="D" THEN D=1
960 IF X$="Z" THEN D=39
970 IF X$="X" THEN D=40
980 IF X$="C" THEN D=41
990 RETURN
1000 CN=32
1010 N1=5
1020 NR=N1
1030 GOSUB 2360
1040 D=0
1050 CM=24
1060 CR=90
1070 CJ=81
1080 J=1024
1090 M=54272
1100 JC=0
1200 RC=0
1210 RY=0
1220 RX=0
1230 R1=0
1240 R2=0
1250 RR=0
1260 MC=0
1270 NM=60
1980 POKE 53280,6
1990 POKE 53281,6
2010 FOR I=0 TO 39
2020 POKE 1064+I,CM
2030 POKE 1064+M+I,MC
2040 POKE 1984+I,CM
2050 POKE 1984+M+I,MC
2060 NEXT I
2070 FOR I=1 TO 22
2080 POKE 1064+I*40,CM
2090 POKE 1064+M+I*40,MC
2100 POKE 1103+I*40,CM
2110 POKE 1103+M+I*40,MC
2120 NEXT I
2130 FOR I=1 TO NM
2140 GOSUB 3000
2150 POKE P,CM
2160 POKE P+M,MC
2170 NEXT I
2180 FOR I=1 TO NR
2190 GOSUB 3000
2200 R(I)=P
2210 POKE P,CR
2220 POKE P+M,RC
2230 NEXT I
2240 GOSUB 3000
2250 J=P
2260 POKE J,CJ
2270 POKE J+M,JC
2280 J1=J
2290 FOR I=1 TO 5
2300 POKE J,CJ+128
2310 FOR K=1 TO 100
2320 NEXT K
2330 POKE J,CJ
2340 FOR K=1 TO 100
2345 NEXT K
2350 NEXT I
2355 RETURN
2360 PRINT CHR$(144);
2370 PRINT CHR$(147);
2380 PRINT "PUNTOS[1SPC]:";S,
2390 FOR I=1 TO NH
2400 PRINT "H";
2410 NEXT I
2420 RETURN
3000 P=INT(RND(TI)*960)+1064
3010 IF PEEK(P)<>32 THEN 3000
3020 RETURN
```

Conexiones



Habiendo cubierto ya las principales etapas de construcción de nuestro brazo-robot, sólo nos resta efectuar las conexiones eléctricas

Paso 1: Conexiones de la veroboard
Cada servomotor tiene tres cables unidos a él. En los servomotores Futaba que recomendamos para este proyecto, los colores de estos cables son: blanco para la línea de control, rojo para la línea de potencia positiva y negro para la línea de retorno común. Los cables del servomotor se deben unir al ordenador controlador y a la fuente de alimentación utilizando un rectángulo de veroboard de 20 franjas ×9 aguj., que se montará en la parte posterior de la caja base del brazo. Corte la veroboard al tamaño adecuado y taladre dos agujeros para el montaje en las posiciones que se indican. Utilice la veroboard a modo de plantilla para marcar las posiciones de los agujeros en la parte posterior de la caja base y haga las perforaciones. Los agujeros de la caja base y la veroboard se deben taladrar de modo que a través de ellos se pueda colocar un tornillo para metales. Haga otro agujero en la parte posterior de la caja base y haga pasar a través del mismo el grupo de cables del servomotor montados dentro de la base. Una entre sí los cuatro grupos de cables de los servos a la parte posterior de la caja base y suelde los cables en las posiciones que se indican. Las cuatro líneas de datos y la línea a tierra del ordenador se deben soldar en la parte de abajo de la veroboard. El diagrama ilustra un cable plano de 20 vías que se utiliza con el BBC Micro. Para el Spectrum y el Commodore se emplean cinco

Paso 1: Conexiones de la veroboard

SERVOMOTOR DEL CUERPO PRINCIPAL
SERVOMOTOR DEL BRAZO INFERIOR
SERVOMOTOR DEL BRAZO SUPERIOR
SERVOMOTOR DE LA PINZA
CARA SUPERIOR DE LA VEROBOARD
AGUJEROS PARA MONTAJE
DIRECCIÓN DE LAS FRANJAS DE COBRE
D0 D1 D2 D3 T
+
–
FUENTE DE ALIMENTACIÓN CD 5 V
CABLE PLANO DE 20 VÍAS (VERSIÓN PARA BBC MICRO)

Paso 2: Conexion

DO

Adaptador de interface para el Spectrum

Conector de la puerta para el usuario del BBC Micro

CONECTOR IDC DE 20 VIAS

Conector de la puerta para el usuario del Commodore 64

TOP

CONECTOR MARGINAL DE 24 VÍAS AL ORDENADOR



cables separados, o un trozo de cable plano de cinco vías. Observe que las líneas de control blancas de los servomotores se sueldan en posiciones que se hallan directamente encima de cada una de las líneas de datos del ordenador, y que todas las líneas negras se sueldan en una única franja de cobre, encima de la conexión a tierra del ordenador y las dos conexiones de la fuente de alimentación. La tarea final de soldadura consiste en fijarle a la placa una fuente de alimentación de 5 V, como se observa en la ilustración. La fuente de alimentación de 5 V de que disponen el BBC Micro, el Spectrum o el Commodore 64 no es apropiada para alimentar los cuatro servomotores con una carga pesada. Debería buscarse, entonces, una fuente alternativa. Una pila de 4,5 V (o tres pilas de 1,5 V unidas con un sujetador de pilas) constituye una fuente de alimentación ideal. Si dispone de un transformador CD de 5 V, también lo puede utilizar. Si pretende emplear pilas, entonces debe soldar un sujetador de pilas en los extremos libres de las líneas de potencia que se extienden desde la veroboard.

Si usted utiliza un transformador, debe instalar un conector de potencia en línea adecuado. El lado negativo de la fuente de alimentación comparte la misma franja de cobre con la conexión a tierra del ordenador y las líneas de retorno negras de los servomotores; el cable de potencia positivo se conecta a cada uno de los cables rojos de los servomotores a través de una franja de cobre común.

Paso 2: Conexiones puerta ordenador

Después de realizar el cableado de la veroboard, debe añadirse el conector adecuado a la puerta del ordenador a los extremos libres de las líneas de datos de D0 a D3 y la línea de tierra. Para el BBC Micro debe utilizarse un cable de 20 vías estándar y un conector IDC. El Commodore 64 emplea un conector marginal de 0,15 pulgadas y 24 vías. Debido a que este conector se enchufa en la puerta para el usuario hacia arriba o hacia abajo indistintamente, antes de comenzar marque uno de los lados como ARRIBA. Las cinco líneas de la veroboard deben soldarse tal como se indica. Tanto el BBC Micro como el Commodore 64 poseen un sistema de circuitos incorporado para tratar las aplicaciones de control a través de una puerta para el usuario. El Spectrum, sin embargo, carece de un sistema de circuitos de este tipo, y debe construirse una interface especial para enchufar en su puerta de ampliación. En anteriores capítulos ya hemos diseñado y construido una interface de este tipo. Ahora los usuarios de un Spectrum habrán de remitirse a aquellos capítulos para construirse la interface. Un conector tipo D de 12 vías que se enchufa directamente en el robot proporciona las líneas de datos y de potencia de la interface. Podemos adaptar este conector para utilizarlo con el brazo-robot. Confeccione un enchufe tipo D de 15 vías empleando los extremos libres de las líneas de datos y tierra provenientes del trozo de veroboard del brazo, y añádale una cubierta para enchufe. Este adaptador permite enchufar el brazo-robot en la puerta para ampliación del Spectrum, a través de la placa de interface.

Paso 3: Montaje del tablero

Cuando haya terminado todo el cableado y lo haya probado cuidadosamente, debe montar la veroboard en la parte posterior de la caja base utilizando dos tornillos para metales ajustados mediante tuercas dentro de la base. Asegúrese de que los motores queden conectados a la veroboard en el orden correcto. La línea de datos D0 (la más alejada hacia la izquierda) debe conectarse a la línea blanca del servomotor del cuerpo principal, montado en la caja base; D1 debe conectarse al servomotor del brazo inferior, montado a la izquierda del cuerpo principal; D2 controla el servomotor del brazo superior, montado junto a la juntura del hombro, y D3 controla el servomotor de la pinza, montado a la derecha del cuerpo principal.

Kevin Jones

H

HACIENDO REGISTROS

Muchos DBM poseen un lenguaje incorporado que permite efectuar búsquedas y acceder a la información con un mínimo esfuerzo

Las bases de datos no se limitan sólo a organizar datos como un conjunto de registros en un archivo. Es posible acceder a los registros mediante instrucciones sencillas, pero los mejores DBM permiten que los usuarios escriban procedimientos empleando un lenguaje de programación incorporado.

Para comprender el empleo en las bases de datos de procedimientos escritos por el usuario, debemos primero repasar la forma en que utilizan las instrucciones la mayoría de los DBM. Una instrucción, como podría ser SEARCH, LOCATE, DISPLAY, NEXT, LAST, etc., es procesada por el DBM para averiguar lo que desea el usuario. Habiendo hecho esto, abre el archivo de la base de datos y lo revisa secuencialmente, comenzando por el primer registro y recorriéndolo todo, hasta el último, extrayendo por el camino aquellos registros que satisfagan las exigencias. Se suele decir que tales instrucciones pertenecen a un "lenguaje de interrogación" (*query*).

En claro contraste con las instrucciones directas o interrogaciones, muchos DBM (dos buenos ejemplos de éstos son *Archive*, de Psion, y *dBase II*, de Ashton Tate) permiten que el usuario escriba programas o procedimientos para simplificar el proceso de extracción o confrontación de la información. Estos procedimientos no son meros conjuntos de instrucciones o interrogaciones estándares. Están escritos en lenguajes de programación completos (si bien limitados) que están incorporados como parte del DBM. Algunos de estos lenguajes de programación de DBM son bastante similares a los de programación normales, y son bastante fáciles de aprender. El *Archive* incorpora un lenguaje que es idéntico al SuperBASIC de Sinclair. Escribir procedimientos para utilizar con *Archive* es tarea sencilla para quien esté familiarizado con el SuperBASIC.

Si bien el lenguaje dBase II no se parece exactamente a ningún otro lenguaje, guarda suficiente parecido a un BASIC estructurado con unas pocas palabras reservadas fáciles de aprender y utilizar. Sus estructuras son bastante sencillas, e incluyen DO WHILE...ENDDO, DO CASE...ENDCASE e IF...ENDIF. Además hay alrededor de un centenar de instrucciones, algunas familiares y otras hechas a medida específicamente para emplear en un entorno de base de datos. Las instrucciones familiares incluyen CALL (para llamar a código máquina), NOTE (equivalente al REM del BASIC), STORE<expresion>to<variable> (equivalente al LET<variable>=<expresion> del BASIC) y muchas otras. Las instrucciones menos familiares incluyen SKIP (para ir hacia adelante o hacia atrás a través de los registros del archivo), PACK (para suprimir del archivo registros no deseados) y FIND<serie de caracteres>.

También se proporcionan otras varias funciones, incluyendo algunas familiares similares al BASIC y otras más inusuales. Ejemplos de ellas son CHR<expresion> (equivalente a CHR$(x)), LEN<expresion de serie de caracteres> (equivalente al LEN del BASIC), TYPE<expresion> (que devuelve el tipo de expresión) y DATE() (una variable del sistema que contiene la fecha).

Para apreciar lo útil que puede ser un procedimiento escrito a la medida, en comparación con el simple uso de una secuencia de instrucciones desde el teclado para extraer la información, emplearemos *dBase II* para crear una base de datos de stock e inventario, y luego escribiremos un procedimiento sencillo en lenguaje dBase II para extraer cierta información. Los registros del archivo serán bastante simples, con cuatro campos numéricos y un campo de caracteres. Un registro típico (también utilizado en el capítulo anterior) sería así:

NUMERO COMPONENTEFABRICANTE	06116
NUMERO COMPONENTENUESTRO	3995
CANTIDAD EN STOCK	86
PRECIO	34.75
DESCRIPCION	Puerta delantera SEAT 127

Primero hemos de hallar un nombre de una sola palabra para cada campo y decidir las longitudes de los campos y si han de ser campos *de caracteres, numéricos* o *lógicos*. Esto sería suficiente:

COMPONENTEFABRICANTE	: 5 carac.; numérico
COMPONENTENUESTRO	: 5 carac.; numérico
EXISTENCIAS	: 3 carac.; numérico
PRECIO	: 6 carac.; numérico
DESCRIPCION	:40 carac.; de caracteres

Una vez puesto en funcionamiento, *dBase II* responde con un punto de requerimiento en la pantalla que indica que está esperando una entrada. La instrucción para iniciar un nuevo archivo es CREATE; también deberemos proporcionar un nombre para el archivo en su conjunto (utilizaremos COMPONENTES como nombre del archivo). El proceso sería algo así:

```
.create componentes <CR>
ENTER RECORD STRUCTURE AS FOLLOWS:
FIELD  NAME, TYPE, WIDTH, DECIMAL PLACES
(entrar estructura del registro del siguiente modo:
campo  nombre, tipo, anchura, posiciones deci-
       males)
001    componentefabricante,n,5,0
002    componentenuestro,n,5,0
003    existencias,n,3,0
004    precio,n,6,2
005    descripcion,c,40
006    <CR>
```

Nuestra entrada se ha dado en letras minúsculas y la respuesta de *dBase II* se ha impreso en letras mayúsculas. Observe que los campos numéricos requieren que, además de la anchura del campo, se especifique el número de posiciones decimales. Pulsando la tecla Return sin entrar ninguna información, como en el campo 006, se termina la fase inicial de creación de un archivo.

A continuación hemos de entrar algunos datos. Esto se realiza utilizando la instrucción APPEND. Ésta hace que se limpie la pantalla y aparezca un "esqueleto" del registro, a la espera de que se entren datos.

```
COMPONENTEFABRICANTE  :          :
COMPONENTENUESTRO     :          :
EXISTENCIAS           :     :
PRECIO                :            :
DESCRIPCION           :
```

Ahora se pueden digitar los datos en el teclado, y se los pasará a cada campo. Un retorno de carro señala que la entrada de un campo ha terminado; apenas se pulse Return para el campo final, se aceptarán todos los datos y volverá a aparecer el esqueleto del registro, esperando detalles para el siguiente registro. Para terminar la entrada de datos se pulsa Return al comienzo de un registro vacío.

Habiendo entrado los datos en el archivo, usted deseará utilizarlos. Supongamos que, para efectuar una verificación contable, necesitara saber el valor de venta al por menor de todo su stock. Una forma en que se podría hacer esto sería ir mirando cada registro de uno en uno en la pantalla y tomando nota de la cantidad de componentes que hay en stock de cada artículo, con el precio de cada uno, y multiplicando las dos cifras. Tras examinar todos los registros y apuntar los precios y cantidades de cada artículo del stock, podría obtener el valor total multiplicando cada precio por el número de componentes y sumando después todos los subtotales. Si estuviera utilizando un fichero de tarjetas convencional, ésta sería la única forma de llegar a la solución.

La mayoría de la DBM permiten recuperar esta clase de información de modo mucho más sencillo. He aquí cómo se podría hacer con *dBase II*:

```
.use componentes
.sum existencias*precio
37870.58
```

La primera línea es una instrucción que solicita a *dBase II* que trabaje con el archivo llamado COMPONENTES. La segunda línea es una instrucción que significa "sumar entre sí los resultados obtenidos tras multiplicar el campo EXISTENCIAS por el campo PRECIO de cada uno de los registros". La tercera línea es la clase de respuesta que podría devolver para el valor total de todo el stock. Muchísimo mejor que los ficheros de tarjetas, ¿verdad?

Este ejemplo ilustra claramente cuán eficaz puede ser un buen DBM, incluso aunque no se utilice más que una sola instrucción. Pero su verdadera eficacia se observa al almacenar secuencias de instrucciones en forma de programa. Para almacenar nuestras sencillas instrucciones de valoración del stock en la forma de un programa denominado VALOR, se necesitaría el siguiente diálogo:

```
.modify command
ENTER FILE NAME: valor
set talk off
use componentes
sum existencias*precio
set talk on
cancel
```

Este corto programa se almacenará en disco y podrá ser empleado desde *dBase II* en cualquier momento digitando la instrucción:

```
.do valor
```

El programa se leerá y se ejecutará automáticamente y, apenas concluido, el control retornará a *dBase II*.

Procedimiento contable

```
*Visualizar contables
SET TALK OFF
USE SOCIOS
DO WHILE.NOT.EOF
  IF PROFESION="CONTABLE"
    DISPLAY NOMBRE
  ENDIF
  SKIP
ENDDO
```

El programa que ofrecemos, que extrae los nombres de los socios de un club que sean contables, está escrito en el lenguaje de instrucciones dBase II de Ashton Tate. La primera línea, que comienza con un asterisco, indica que se trata de una línea de comentarios. No requerimos que se visualicen todos los números de registros mientras *dBase II* va efectuando la búsqueda, de modo que interrumpimos la conversación (SET TALK OFF). En la base de datos podría haber retenidos varios archivos diferentes, de modo que debemos especificar en qué archivo (en este caso, SOCIOS) se ha de llevar a cabo la búsqueda. Puesto que deseamos buscar a través de todos los registros disponibles, construimos un bucle alrededor de la estructura DO WHILE...ENDDO. Dentro del bucle establecemos una condición que afirma que siempre que el campo de la profesión contenga la serie "CONTABLE", el programa debe visualizar el campo del nombre del registro. Observe que la condición IF debe concluir con un ENDIF. El programa ejecuta entonces SKIP, que le indica a *dBase II* que pase al siguiente registro



Descubriendo nuevos mundos
El Discovery 1 está considerado como un sistema de ampliación "todo en uno" para el usuario del Spectrum. La máquina viene completa con una unidad de disco, sistema operativo y conexiones que permiten la instalación de impresoras, palancas de mando, monitores de vídeo compuesto y otros periféricos, sin necesidad de interfaces adicionales

Sistema total

Con el examen del Discovery 1, de Opus, finalizamos nuestra serie dedicada a las alternativas al microdrive de Sinclair

En el capítulo anterior de este apartado analizamos el Wafadrive de Rotronics. Si bien este dispositivo resultó ser algo más fiable que el microdrive, en comparación es más lento y basado todavía en el sistema, más bien sospechoso, de cinta continua. En este capítulo centramos nuestra atención en un enfoque más convencional al almacenamiento masivo: un sistema basado en disco de Opus Supplies.

Una unidad de disco para el Spectrum no constituye ninguna idea nueva. Durante los dos últimos años han aparecido diversos sistemas operativos de disco e interfaces para la máquina; sin embargo, ninguno de estos sistemas se ha hecho especialmente popular. Ello se debe en parte a que las interfaces se han promocionado sólo en la prensa especializada, lo que puede haber dado la impresión de que los dispositivos estaban destinados sólo al entusiasta del Spectrum y a los programadores de lenguaje máquina, y sólo se han puesto a la venta a través de casas de venta por correspondencia y no se les ha dado el tipo de comercialización masiva necesaria para introducirlos en el público.

El Discovery 1 viene en una carcasa de chapa metálica, con un frente que se prolonga hacia adelante para soportar al Spectrum, que se enchufa en una ranura para cartuchos a través de la puerta para ampliación. Aunque la operación de este sistema parece comparativamente sencilla, quizá los usuarios encuentren dificultades para instalar el Discovery en el conector marginal. Ello se debe a que los cables de la cassette y la antena tienden a entorpecer la operación dificultando la correcta fijación en la ranura para cartuchos. Éste es un problema grave si se considera que un conector marginal colocado incorrectamente podría dañar gravemente al Spectrum. La dificultad no es tan pronunciada como con el Spectrum original (para el cual se diseñó originalmente el Discovery), pero el nuevo Spectrum+, con su carcasa más grande, puede exigir que los usuarios se esfuercen durante varios minutos antes de quedar satisfechos con la instalación correcta del dispositivo. Una vez que el Spectrum se ha instalado correctamente en el Discovery 1, la máquina en realidad bloquea la propia entrada de la fuente de alimentación del ordenador. El Discovery, por consiguiente, ha sido diseñado para alimentarse a sí mismo y al Spectrum, haciendo innecesaria la fuente de alimentación externa del micro.

Encima de la ranura para cartuchos de la izquierda hay una unidad de disco simple de 3 ½ pulgadas, con espacio a la derecha para una segunda unidad. (Opus tiene la intención de lanzar una versión de la máquina con unidad doble, denominada Discovery 2.) Los usuarios del Discovery 1 que deseen mejorar su sistema convirtiéndolo en un sistema de doble unidad, pueden aguardar el lanzamiento de una unidad de disco adicional que se llamará Discovery+. No obstante, los usuarios no se verán limitados sólo a las unidades de Opus, dado que la empresa afirma que también se pueden añadir unidades de 5 ¼ pulgadas.

Al igual que Rotronics, la filosofía de Opus al diseñar la máquina es la de proporcionar no sólo un sistema de almacenamiento masivo para el Spectrum, sino también añadir interfaces extras para periféricos que permitan a los usuarios el empleo de impresoras y otros dispositivos. En la parte poste-



Carcasa
Hemos dado la vuelta a la carcasa del Discovery 1 para poder apreciar la posición de las interfaces para periféricos

Puerta para palanca de mando
El Discovery 1 lleva instalado un único conector tipo Atari, tipo D de 7 patillas, que permite la conexión de una palanca de mando

Puerta para impresora
Es una interface en paralelo compatible con Centronics, susceptible de utilizar con la mayor parte de impresoras

Conector para periféricos
Para permitir la futura ampliación del sistema, aquí se duplica el conector marginal del Spectrum

Chip de interface para periféricos
Se utiliza el chip 6821 para controlar los periféricos

Conector para cartuchos
Aquí se instala la ranura para ampliación del Spectrum, para conectarlo en interface con el Discovery

Chips de RAM
El Discovery posee su propia RAM incorporada, de modo que para el DOS se utiliza la mínima cantidad posible de memoria del ordenador

Conector vacío
Se proporciona para retener un chip de RAM de 2 K, que sería necesario si se añadiera una segunda unidad

ROM del sistema operativo
El DOS del Discovery está contenido en una EPROM de 8 K

Transformador
El transformador se ha colocado directamente debajo de la unidad, lo que ha causado algunos problemas de recalentamiento

Unidad de disco
El Discovery utiliza discos de 3 ½ pulgadas de formato Sony. A uno de los lados hay un espacio libre que permitirá al Discovery 2 incluir una unidad adicional

DISCOVERY 1

DIMENSIONES
300×210×75 mm

INTERFACES
Conector paralelo, interface en paralelo Centronics, puerta para palanca de mando, enchufe para video compuesto

FORMATO
Discos estándares Sony de 3 ½ pulgadas, de doble densidad y doble cara

CAPACIDAD
250 K total, 180 K tras el formateado

VELOCIDAD
15 Kbaudios de velocidad de transferencia; tiempo de acceso de pista a pista: 3 milisegundos

rior del Discovery 1 se proporciona un conector para monitor de video compuesto, según declara un portavoz de Opus, para los usuarios de gestión que deseen incorporar una pantalla monocromática (aunque, por supuesto, el video compuesto sí produce una señal de color) para prolongados períodos de tratamiento de textos. No obstante, es de lamentar que en una máquina que destaca por sus coloridos programas de juegos Opus no haya podido proporcionar una interface RGB para producir una imagen mucho más nítida.

En el lado derecho del Discovery hay una puerta para una palanca de mando tipo Atari compatible con Kempston, junto a la cual hay una puerta para impresora en paralelo Centronics bidireccional. Por último, hay un conector para periféricos que permite la conexión de otras interfaces compatibles con el Spectrum, como un monitor RGB.

Al igual que en el Wafadrive, el sistema operativo de disco del Discovery es muy similar al de la Interface 1; por ejemplo, para generar una instrucción se requiere <INSTRUCCION>*. Cuando el intérprete de BASIC llega al *, no lo reconoce como si se tratase de una instrucción de BASIC e intenta generar un mensaje de error de sintaxis. No obstante, el DOS lo intercepta y lo envía a su propio sistema operativo de 8 K en la posición de los 8 K inferiores de la ROM del Spectrum, y luego interpreta la instrucción. Hay que destacar que si el usuario hubiera realizado un error de sintaxis, el DOS no lo reconocería y se generaría un mensaje de error, si bien ello se haría a través de la ROM del DOS.

Al diseñar su sistema DOS Opus ha ido más lejos que Rotronics al proporcionar compatibilidad: se han conservado todas las instrucciones disponibles en el microdrive. Esto obedece a varios motivos. Debido al sistema de entrada del Spectrum, de una única palabra clave, obviamente es más fácil escribir un sistema operativo que utilice instrucciones inherentes, en vez de meterse en el problema de escribir instrucciones propias. Esto también significa que los usuarios que ya estén familiarizados con el sistema operativo del microdrive podrán utilizar el Discovery de inmediato, puesto que toda la sintaxis es la misma. Además, trabajar con un sistema operativo puede llevar a todo tipo de problemas imprevistos con el mapa de memoria. Ello significa que los programas compatibles con la Interface 1 podrían no ser necesariamente compatibles con su sistema operativo remendado, un problema que se ha presentado con asiduidad en muchos periféricos de otras empresas independientes.

La forma en que Opus ha seguido de cerca el sistema de instrucciones del microdrive de Sinclair se hace más evidente cuando se observa la manera en que se han organizado las corrientes. En el Spectrum, los canales de salida están organizados en 16 corrientes numeradas de la 0 a la 15. Tres de éstas están reservadas para la pantalla, el teclado y la impresora, y las otras están libres para utilizarlas con cualquier otro periférico. En la lista de instrucciones de Sinclair para la Interface 1 existen diversos caracteres individuales que abren canales a dispositivos específicos; por ejemplo, 'm' para microdrive. El Discovery ha adaptado las mismas para su propio uso, de modo que la instrucción LOAD*'m-';1;'nombre' funcionará igualmente en el Discovery 1 y en el microdrive, aunque en este caso 'm' alude a la unidad de disco. Sin embargo, por razones de conveniencia, Opus ha adaptado el formato de instrucción de modo que se pueda omitir la 'm', abreviando de este modo el sistema de Sinclair, en cierta forma muy largo. Asimismo, hay otras instrucciones que también se han adaptado. En el microdrive, el carácter 't' en una instrucción abre un canal a la interface RS232, mientras que en el Discovery el mismo carácter abre el canal a la impresora en paralelo.

Los discos en funcionamiento

La unidad de disco que se ofrece con el sistema emplea los discos de 3 ½ pulgadas de formato Sony de doble densidad, que cada día se utilizan con mayor asiduidad en los microordenadores. Cada uno de los discos posee una capacidad total de 180 K de espacio de almacenamiento. El DOS soporta acceso directo cuando se busca en un archivo, lo acceso directo cuando se busca en un archivo, lo que es considerablemente más rápido que los métodos de búsqueda secuencial que utilizan algunos otros sistemas de disco. Asimismo, no existe límite alguno en cuanto a la cantidad de archivos que se pueden retener en un disco, lo que puede ser importante cuando se desea guardar numerosos archivos cortos. Si el directorio se llenara rápidamente, podría quedar en el disco una gran cantidad de espacio que no se podría aprovechar.

Al comparar el tiempo que lleva guardar (SAVE) y cargar (LOAD) un archivo utilizando el Discovery y el microdrive, el primero resultó algo más rápido para encontrar el archivo, pero considerablemente más lento para guardarlo y cargarlo. Hallar un archivo es más rápido en el sistema de disco porque los archivos están organizados por acceso directo, mientras que los microdrives, por su propia naturaleza, son dispositivos de acceso secuencial. Las razones por las cuales cargar un archivo en la memoria es mucho más lento resultan difíciles de explicar, pero la realidad es que la velocidad de transferencia del Discovery 1 es inferior a la de los microdrives: 15 Kbaudios frente a 19,2 Kbaudios. La verdadera ventaja del Discovery radica en tener un sistema de almacenamiento masivo más eficaz que el del microdrive, y un medio de almacenamiento que dispone de una gama de fabricantes más amplia.

Opus parece haber pensado muy bien en el problemático aspecto del apoyo de software para el Discovery. Obviamente, contar con una gran empresa que venda el producto en su cadena de establecimientos es una ventaja, porque las empresas de software pueden ofrecer sus productos junto con la propia máquina. La empresa también ha señalado que muchas casas de software ya se han comprometido a transferir algunos de sus programas existentes a los discos de 3 ½ pulgadas.

El lanzamiento de la serie de unidades de disco Discovery ha sido evidentemente muy bien planeado y Opus, como es lógico, ha intentado proporcionar a su nueva línea las mayores probabilidades de éxito posibles. La tarea primordial que tiene la empresa ante sí es convencer a los usuarios del Spectrum de que el Discovery constituye una inversión más valiosa para sus máquinas que la que representa la alternativa de Sinclair.

Llaman a la puerta

Vamos a realizar un estudio de las entradas y salidas tratadas por el sistema operativo del Commodore 64

La fotografía que adjuntamos muestra todas las puertas disponibles en la parte trasera del Commodore 64. Junto a la puerta para cassette, para audio/video y para la TV, existen tres puertas de E/S a disposición del programador. Son, de izquierda a derecha, la puerta para ampliación (también denominada puerta para cartucho) la puerta serial (o bus serial) y la puerta para el usuario. Veámoslas por separado.

• *Puerta serial*: Las impresoras seriales Commodore y la unidad de disco serial 1541 se conectan mediante el enchufe DIN de seis patillas. La instrucción OPEN hará siempre referencia a esta puerta con cualquier número de dispositivo salvo el 2 (especificar el dispositivo 2 significa la RS232 por medio de la puerta para el usuario). Por ejemplo, con el número de dispositivo 8, la instrucción OPEN2, 8,2,"NOMBREARCHIVO" abrirá un archivo en la unidad de disco y a través de la puerta serial. No es aconsejable hacer uso de la puerta serial por medio de rutinas en BASIC o del núcleo, salvo con dispositivos Commodore.

Se pueden adquirir interfaces que aceptan mensajes seriales a través de esta puerta y realizan una conversión paralelo serial/IEEE. Esto permite que los dispositivos periféricos sean conectados al PET Commodore y puedan ser empleados en el Commodore 64. Tales periféricos incluyen las unidades de disco Commodore 4040.

• *Puerta para el usuario*: Es un conector plano de 24 vías, macho, de 0,15 pulgadas, utilizable tanto para la comunicación en serie como en paralelo. Por ejemplo, esta puerta puede servir para conectar el Commodore 64 a una impresora no Commodore (como la Epson, que es la más frecuente), la cual es tratada como un dispositivo RS232. Ya analizaremos más adelante cómo se puede manipular la RS232 desde el OS.

La puerta para el usuario puede también emplearse en la comunicación en paralelo de 8 bits, pero ha de escribirse (o adquirirse) el software específico para gestionar dispositivos, ya que el OS no posee rutinas de control, o *drivers* (código máquina para control de periféricos), para la comunicación en paralelo. En el próximo capítulo ofreceremos un programa titulado Parawedge, que ilustra el trabajo global necesario para obtener un programa de comunicaciones en paralelo por medio de interrupciones con un grado de sofisticación mediano. Este programa podrá servir para transferir un bloque de memoria desde el ordenador a algún dispositivo externo (p. ej., trasladar un programa en BASIC de un Commodore 64 a otro), siempre que ambas máquinas tengan niveles de 5 V.

• *Puerta para ampliación*: Se trata de una entrada hembra de 44 patillas que proporciona el acceso a todas las líneas de control principal, dirección y datos del Commodore 64. Esta puerta sirve para acoplar cartuchos de juegos y cartuchos de IEEE en paralelo, responsables de la conexión del ordenador con periféricos PET. Sirve igualmente para cualquier otro dispositivo o software que requiera casi un completo control de la máquina o programas suficientemente aptos para garantizar el diseño y la construcción de una tarjeta para retenerlos como ROM o EPROM.

Proporcionamos unos diagramas de cada una de estas tres puertas mostrando las funciones de sus conexiones de patillas. Volvamos ahora a ocuparnos de aspectos del empleo que hace el sistema operativo de estas puertas. El conjunto de rutinas OS que se encargan de E/S se conoce como núcleo. Son programas en código máquina invocados por las sentencias en BASIC de E/S tales como OPEN, CLOSE, GET#, PRINT#. La organización de las rutinas del núcleo es tal que el programador puede acceder a ellas con facilidad a través de sistemas en lenguaje máquina. Damos un breve listado del lenguaje máquina que permite el uso de la rutina del núcleo LOAD.

Comunicación en paralelo

También vamos a examinar aquí la capacidad RS232 incorporada en el Commodore 64. Consideraremos además cómo se usa, por ejemplo, para tratar un módem por medio de las rutinas RS232.

Pero no siempre será posible el empleo de rutinas de sistema para la E/S. A veces puede ser necesario el empleo de una comunicación en paralelo y grandes cantidades de datos para un dispositivo propio cercano, más que una comunicación en serie con algún dispositivo remoto. No existen rutinas de sistema para el manejo de la puerta para el usuario en el Commodore 64. Por ello la comunicación en paralelo implica la programación de los dos Adaptadores de Interface Complejos 6526 (CIA). Para iniciarle en ello, daremos en el próximo capítulo una rutina en ensamblador de transferencia de 8 bits de datos en paralelo. Tal rutina permite al ordenador leer o enviar datos por medio de la puerta para el usuario mientras se procesa al mismo tiempo un programa en BASIC. El malabarismo del tiempo compartido se logra haciendo que el código que lee o escribe datos acepte interrupciones. En el capítulo anterior de esta serie dimos un ejemplo de una cuña IRQ (requerimiento de interrupción). Aquí la rutina acepta una NMI ya que la línea de flag de la puerta para el usuario puede servir para generar una NMI (interrupción no enmascarable): es la segunda línea de interrupción para el procesador 6510. A diferencia de la IRQ, no se puede enmascarar una señal de interrupción en la línea NMI por medio de las instrucciones SEI y CLI.

Puertas de E/S en el Commodore 64

Parte superior

Patilla	Tipo	
1	GND	Tierra
2	—	+5 voltios a un máximo de 100 mA
3	—	RESET del sistema
4	CNT1	Contador de puerta serial CIA#1
5	SP1	Puerta serial CIA#1
6	CNT2	Contador de puerta serial CIA#2
7	SP2	Puerta serial CIA#2
8	PC2	Línea "apretón de manos" desde CIA#2
9	—	Esta línea se conecta con ATN en la puerta serial
10	—	Fase + de 9 V ac
11	—	Fase − de 9 V ac
12	—	Tierra

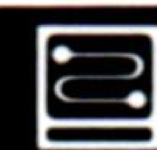
Notas:

1) Línea-3=protocolo X on/X off
Línea-X=protocolo CTS/RTS

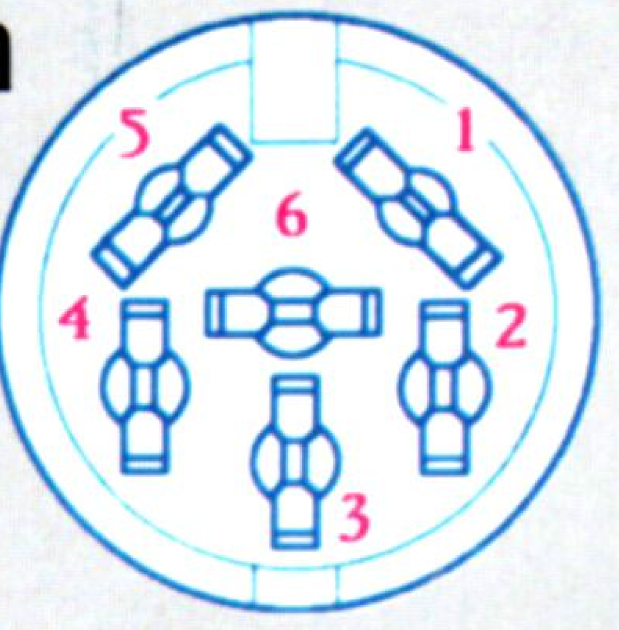
2) (*) Además de ser direccionadas por medio de las rutinas del núcleo RS232 del OS, estas líneas pueden también programarse directamente para E/S definidas por el usuario. Tales conexiones se emplearon en el apartado *Bricolaje* para controlar dispositivos externos como la caja buffer y el robot móvil

Parte inferior

Patilla	Tipo	Función RS232 (*)	Usadas en línea-3/línea-X
A	GND	Tierra protectora	Ambas
B	FLAG2	Datos recibidos —entrada	Ambas
C	PB0	Datos recibidos —entrada	Ambas
D	PB1	Solicitud de envío (RTS) —salida	Ambas (alto en 3-1)
E	PB2	Terminal datos preparado (DTR) —salida	Ambas (alto en 3-1)
F	PB3	Indicador de timbre (RI) —entrada	—
H	PB4	Señal línea recibida (DCD) —entrada	Sólo línea-X
J	PB5	Sin asignar	—
K	PB6	Borrar para envío (CTS) —entrada	Sólo línea-X
L	PB7	Total datos preparado (DSR) —entrada	Sólo línea-X
M	Datos	Datos transmitidos —salida	Ambas
N	GND	Señal tierra	Ambas



Puerta serial



Patilla	Tipo	Observaciones
1	SRQIN Serial	Servicio de solicitud
2	GND	
3	ATN E/S SERIAL	Señal de atención para disp. en puerta
4	CLK E/S SERIAL	Reloj temporizador para puerta serial
5	E/S DATOS SERIAL	Línea de transporte datos de un solo bit
6	RESET	Línea de RESET del hardware

Puerta ampliación



Parte superior

Patilla	Tipo	Observaciones
1	GND	
2	+ 5V	
3	+ 5V	
4	IRQ	Petición de interrupción
5	R/W	Posibilitar lectura/escritura
6	Reloj puntos	
7	E/S 1	
8	GAME	
9	EXROM	
10	E/S 2	
11	ROML	
12	BA	
13	DMA	
14	D7	
15	D6	
16	D5	
17	D4	8 líneas de datos
18	D3	
19	D2	
20	D1	
21	D0	
22	GND	

Parte inferior

Patilla	Tipo	Observaciones
A	GND	
B	ROMH	
C	RESET	Línea de RESET del hardware
D	NMI	Interr. no enmascarable
E	S02	
F	A15	
H	A14	
J	A13	
K	A12	
L	A11	
M	A10	
N	A9	
P	A8	16 líneas de dirección
R	A7	
S	A6	
T	A5	
U	A4	
V	A3	
W	A2	
X	A1	
Y	A0	
Z	GND	

El núcleo en acción

El núcleo (*kernel*) es una biblioteca de rutinas de E/S, que son llamadas por el BASIC o por código máquina escrito por el usuario. La ROM del núcleo está situada de $E000 a $FFFF, pero las rutinas deseadas se llaman desde una "tabla de salto", es decir, desde una tabla de punteros que apuntan a las posiciones de inicio de las rutinas y que se guarda en la parte superior de la memoria. La ventaja de este sistema de llamadas a través de una tabla de salto consiste en la facilidad con que puede emplearse el código máquina incluso con diferentes máquinas Commodore, pues la tabla de salto no varía aunque varíen los contenidos.

Para usar los programas núcleo se necesita material de referencia. La Guía de Referencia para el Programador que proporciona Commodore es quizá la mejor fuente. P. ej., para servirnos de la rutina de núcleo LOAD y realizar con ella un LOAD de reubicación, suele emplearse el siguiente fragmento de código. Primero se colocarán los códigos ASCII (según la versión del Commodore) correspondientes al nombre del archivo en direcciones de memoria consecutivas, y después se hará uso del siguiente código máquina:

```
LDA  #$01    Número lógico del archivo (lfn)
LDX  #$08    Número del dispositivo (disk)
LDY  #$00    Dirección secundaria (#$00 da
             la carga de reubicación)
JSR  $FFBA   Rutina de núcleo SLFS
             (establece lfn y dir. sec.)
LDA  #$0A    Long. nombre archivo
             (p. ej., 10)
LDX  PLO     Byte inferior del puntero a la dir.
             de inicio del nombre archivo
LDY  PHI     Byte sup. del puntero a la dir.
             de inicio del nombre archivo
JSR  $FFBD   Rutina de núcleo SETNAM
             (establecer nombre de archivo)
LDA  #$00    Carga=#$00/Verificación=#$01
LDX  DLO     Byte inf. de la dir. inicio destino
LDY  DHI     Byte sup. de la dir. inicio destino
JSR  $FFD5   Rutina de núcleo LOAD
```

Empleo de la RS232 en el Commodore 64

Siempre que se realice un planteamiento metódico no hay peligros serios en el empleo de rutinas OS para manipular la RS232, ya sea desde el BASIC o desde código máquina. Varios son los aspectos a considerar al aprestarnos a usar la RS232 a través de la puerta para el usuario. Dejando a un lado la velocidad en baudios, de la que volveremos a hablar, los únicos problemas con que se encontrará serán los referidos a estos puntos:

- El Commodore 64 funciona a niveles de 0 V a 5 V, mientras que la RS232 normal necesita niveles de −12 V a 12 V. Luego, a menos que no se comunique con otro Commodore 64, será preciso construir o adquirir un dispositivo para "conversión de niveles". Commodore proporciona un cartucho RS232 para esta circunstancia.
- Los códigos ASCII del Commodore se alejan de los códigos ASCII habituales, por lo que habremos de establecer dos tablas conversoras, una para transmitir y otra para recibir.

Siempre que esté abierto un canal RS232, el OS realiza un borrado automático de registros (CLR). Se pierden así inmediatamente todos los valores de las variables usadas por el BASIC con anterioridad; las instrucciones GOSUB darán mensaje de error al llegar a RETURN. Esto se debe a que la rutina de núcleo de la RS232 ocupa dos buffers de 256 bytes en la parte superior de la memoria. Esto puede ser problemático, ya que si no hay espacio suficiente para los buffers en esa zona, se corromperá el programa y no se tendrán mensajes de error.

- Si el programa en BASIC es largo o contiene gran número de asignaciones alfanuméricas, tarde o temprano obtendrá basura. Pueden perderse datos introducidos. Pero no siempre ha de suceder esto si el programa es de una longitud razonable y no se hacen tantas asignaciones alfanuméricas. Para abrir (OPEN) un canal RS232 desde el BASIC se emplea el siguiente formato:

```
OPEN 2,2,3,CHR$(CTRL)+CHR$(COMM)
```

CTRL y COMM son los bytes de control e instrucción, que contienen la información necesaria para establecer el canal. Nótese que CTRL y COMM deben ser dos bytes literales (o los PEEK de dos posiciones previamente llenadas) y no dos variables. Cada bit dentro de los bytes de CTRL y COMM tiene una función precisa (ver cuadros).

Por ejemplo, para abrir un canal RS232 con un bit de parada, una longitud de palabra de 7 bits, 300 baudios (total del byte de CTRL=38), paridad par, *full duplex* y control de flujo línea-3 (total del byte de COMM=96), emplearemos la instrucción:

```
OPEN2,2,3,CHR$(38)+CHR$(96)
```

Al decidir la velocidad de transmisión no se han de olvidar los siguientes factores. Al enviar información, la velocidad en baudios no es tan esencial, ya que los restantes dispositivos son bastante flexibles en las variaciones de velocidad. Se han enviado con éxito caracteres desde el BASIC a través de la RS232 hasta a 2 400 baudios. Naturalmente, mientras la proporción de bits por byte es de 2 400, el espacio entre bytes es a menudo mucho mayor, de modo que la proporción de transferencia resulta mucho menor.

Pero las cosas cambian cuando se trata de recibir datos. En este caso, incluso a 300 baudios un programa en BASIC apenas tendrá tiempo de captar un byte desde el buffer de entrada y visualizarlo en la pantalla dentro de un bucle simple. Para recibir con efectividad se debe usar algún "control de flujo". Esto significa que se pide al resto de dispositivos que interrumpan su envío hasta que no se llene el buffer. Con el protocolo línea-3 el procedimiento genérico es el siguiente:

1) Leer un pequeño número de bytes (cuanto mayor sea la velocidad de transmisión menor ha de ser este número) empleando GET#2,A$. Tratarlos inmediatamente o colocarlos en una tabla para su posterior proceso.

2) Interrumpir el envío de los otros dispositivos con la instrucción PRINT#2, CHR$(17).

3) Leer más bytes hasta que se vacíe el buffer real de RS232. Procesar cómodamente todos los bytes leídos, mientras se comprueban las teclas pulsadas, señales de final de archivo (EOF), etc.

4) Permitir que el dispositivo envíe de nuevo empleando PRINT#2, CHR$(19) y volver al paso 1.

Cuando se ha abierto un canal RS232, los bytes se envían o reciben de la manera acostumbrada, por medio de PRINT# o GET#. Ha de evitarse el empleo de INPUT#. El byte de estado, ST, debe ser comprobado, lo mismo que exige todo programa con E/S; el estado de error cero se indica por ST=0 o bien ST=8. Por último, CLOSE producirá sin duda otra CLR automática, dado que los buffers son desubicados.

Byte CTRL

Bit 7	6	5	4	3	2	1	0	Función
—	—	—	X	0	0	0	1	50 baudios
—	—	—	X	0	0	1	0	75 baudios
—	—	—	X	0	0	1	1	110 baudios
—	—	—	X	0	1	0	0	134,5 baudios
—	—	—	X	0	1	0	1	150 baudios
—	—	—	X	0	1	1	0	300 baudios
—	—	—	X	0	1	1	1	1 200 baudios
—	—	—	X	1	0	0	0	1 800 baudios
—	—	—	X	1	0	1	0	2 400 baudios
—	0	0	X	—	—	—	—	datos de 8 bits
—	0	1	X	—	—	—	—	datos de 7 bits
—	1	0	X	—	—	—	—	datos de 6 bits
—	1	1	X	—	—	—	—	datos de 5 bits
0	—	—	X	—	—	—	—	1 bit parada
1	—	—	X	—	—	—	—	2 bits parada

X=valor irrelevante

Byte COMM

Bit 7	6	5	4	3	2	1	0	Función
—	—	—	—	X	X	X	0	Protocolo línea-3
—	—	—	—	X	X	X	1	Protocolo línea-X
—	—	—	0	X	X	X	—	*Full duplex*
—	—	—	1	X	X	X	—	*Half duplex*
—	—	0	—	X	X	X	—	Paridad desactivada
0	0	1	—	X	X	X	—	Paridad impar
0	1	1	—	X	X	X	—	Paridad par
1	0	1	—	X	X	X	—	Comprob.-p envío señal desact.
1	1	1	—	X	X	X	—	Comprob.-p envío no señal desact.

mi computer

Editorial Delta, S.A.



10082
9 788485 822836